EXHIBIT 2

UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

CISCO SYSTEMS, INC.

Petitioner

_____

IPR2024-01281
U.S. Patent No. 7,738,471

PETITION FOR _INTER PARTES_ REVIEW
UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104

IPR2024-01281 Petition
Inter Partes Review of U.S. 7,738,471

TABLE OF CONTENTS

IPR2024-01281 Petition
Inter Partes Review of U.S. 7,738,471

PETITIONER'S EXHIBIT LIST

Ex.1001	U.S. Patent No. 7,738,471 to Kuliner
Ex.1002	File History of U.S. Patent No. 7,738,471
Ex.1003	Declaration of Nader F. Mir, Ph.D.
Ex.1004	*Curriculum Vitae* of Nader F. Mir, Ph.D.
Ex.1005	U.S. Patent Appl. No. 2006/0072564 to Cornett et al. ("Cornett")
Ex.1006	U.S. Patent No. 7,333,489 to Nelson et al. ("Nelson")
Ex.1007	U.S. Patent Appl. No. 2006/0209840 to Paatela et al. ("Paatela")
Ex.1008	Transmission Control Protocol, RFC 793 (Sept. 1981)
Ex.1009	Andrew S. Tanenbaum, "Computer Networks," 3rd ed. (1996) ("Tanenbaum")
Ex.1010	U.S. Patent Appl. No. 2006/0215695 to Olderdissen ("Olderdissen")
Ex.1011	U.S. Patent No. 7,821,931 to Swenson et al. ("Swenson")
Ex.1012	U.S. Patent No. 8,396,064 to Giesberts et al. ("Giesberts")
Ex.1013	U.S. Patent No. 9,307,054 to Boucher et al. ("Boucher")
Ex.1014	U.S. Patent No. 6,175,571 to Haddock et al. ("Haddock")
Ex.1015	U.S. Patent No. 6,438,128 to Kashyap ("Kashyap")
Ex.1016	U.S. Patent No. 6,501,764 to Fudatate et al. ("Fudatate")
Ex.1017	U.S. Patent No. 7,110,404 to Temoshenko ("Temoshenko"),
Ex.1018	U.S. Patent Appl. No. 2003/0223376 to Elliott et al. ("Elliott")
Ex.1019	U.S. Patent Appl. No. 2005/0256975 to Kaniz et al. ("Kaniz")
Ex.1020	U.S. Patent. No. 6,650,654 to Batty et al. ("Batty")

Ex.1021	Interim procedure for Discretionary Denials in AIA Post-Grant Proceedings with Parallel District Court Litigation (June 21, 2022)
Ex.1022	Complaint filed in *Lionra Technologies Ltd. v. Cisco Sys., Inc.*
Ex.1023	Radia Perlman, "Interconnections: Bridges, Routers, Switches, and Internetworking Protocols," 2d ed. (1999) ("Perlman")
Ex.1024	U.S. Patent Appl. No. 2002/0003795 to Oskouy et al. ("Oskouy")
Ex.1025	U.S. Patent No. 6,678,746 to Russell et al. ("Russell")
Ex.1026	WO 00/10302 to Lawton et al. ("Lawton")

I.   INTRODUCTION

Pursuant to 35 U.S.C. §§311, 314(a), and 37 C.F.R. §42.100, Cisco Systems, Inc. ("Petitioner") respectfully requests that the Board review and cancel as unpatentable under (pre-AIA) 35 U.S.C. §103(a) claims 1-3, 5-10, 12-15, 17-21 ("Challenged Claims") of U.S. 7,738,471 ("'471 patent," Ex.1001).

The '471 patent is directed to high-speed packet processing. Ex.1001, Title. The claims purport to reduce packet processing time by using a direct memory access ("DMA") device to write each header of a packet to a packet buffer memory and, concurrently, write each header to a respective protocol stack layer memory for processing. Ex.1001, 5:64-6:6. Such concepts were well known before the '471 patent's earliest priority date. For example, Cornett discloses a packet processing technique that uses a DMA device to write packets and packet headers into separate memory locations, Ex.1005, [0081], Paatela explains how packet processing can be accelerated by writing each individual packet header to a separate memory location, Ex.1007, [0077], [0079], and Nelson teaches writing packets and packet headers into separate memory locations in parallel so that routing decisions can be made more efficiently. Ex.1006, 2:1-10, 7:1-4. Russell further improves packet processing by separating packet headers from data and then processing the headers in parallel. Ex.1025, claim 8 at 7:29-8:10.

Because the Challenged Claims merely recite an obvious combination of known concepts, Petitioner asks the Board to institute trial and find the claims unpatentable.

II.   GROUNDS FOR STANDING

Petitioner certifies that the '471 patent is eligible for IPR and that Petitioner is not barred or estopped from requesting IPR of the Challenged Claims.

III.   NOTE

Petitioner cites to exhibits' original page numbers. **Emphasis** in quoted material has been added. Claim terms are presented in *italics*.

IV.   TECHNICAL BACKGROUND

"Understanding, designing, and building a computer network would be too difficult a task unless the problem were partitioned into smaller subtasks, traditionally by dividing the problem into layers." Ex.1023, 1. One of the most common set of layers used by networking professionals is the Open Systems Interconnection ("OSI") model, which has seven layers.[1] Ex.1009, 28, 36. This model is shown below:

---

[1] The OSI model is not the only model for network communications. Another common set of layers is the Transmission Control Protocol/Internet Protocol ("TCP/IP") model. Ex.1009, 28.

Figure 1.1 The OSI Reference Model
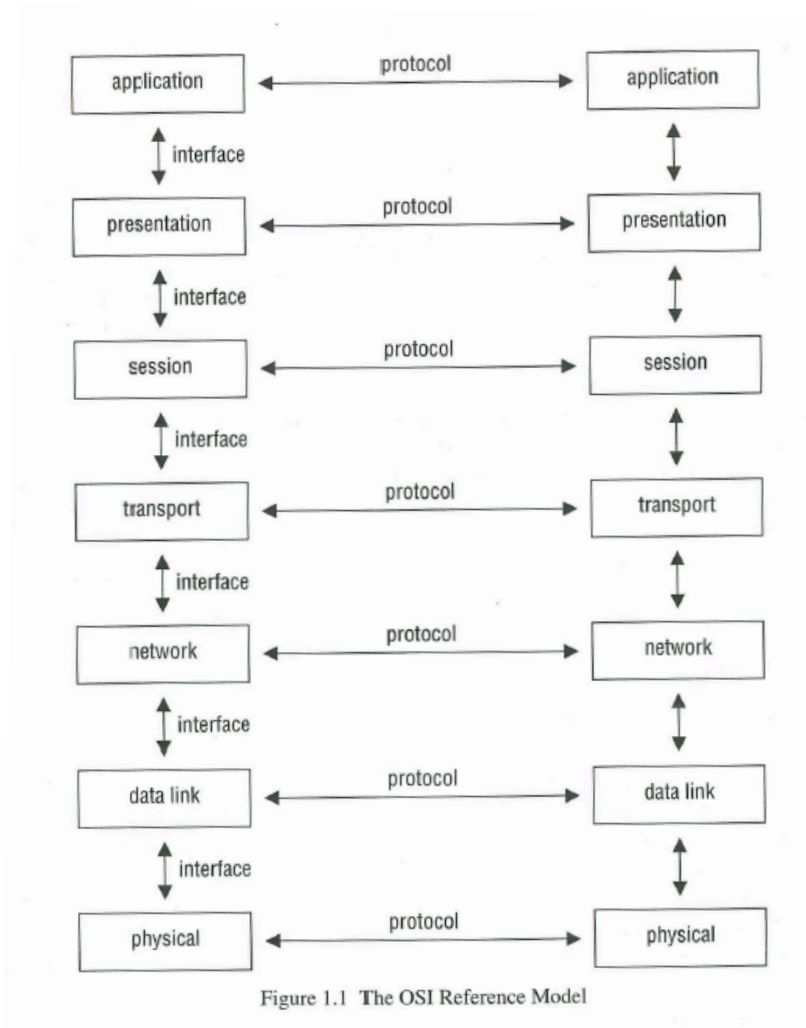
Ex.1023, 2

Each layer at a source network node communicates with its peer layer at a destination network node "through the use of a protocol." Ex.1023, 1. It is common for a single layer to support several protocols. *See* Ex.1023, 3 ("it is common for different links to implement different data link layers and for a node to support several data link layer protocols, one for each of the types of links to which the node is attached."). The topmost layer—the application layer—includes an application running on a network node, such as a web browser or email client.

9

Ex.1023, 4. The bottommost layer—the physical layer—includes the physical link, such as a copper wire, that carries a communication. Ex.1023, 2; Ex.1012, 1:65-67.

A protocol operating at a particular layer receives a packet of information and adds a header to it. Ex.1023, 4. The header acts like an envelope for a letter that helps deliver the letter to its destination. *See* Ex.1023, 4 (for example, network layer adds a header that "includes information such as the source and destination addresses."). As the packet is passed down through the successive layers, protocols operating within those lower layers may add their own headers to the packet. *See* Ex.1009, 19 ("A message, *M*, is produced by an application process running in layer 5 and given to layer 4 for transmission. Layer 4 puts a header in front of the message to identify the message and passes the result to layer 3. The header includes control information, such as sequence numbers, to allow layer 4 on the destination machine to deliver messages in the right order if the lower layers do not maintain sequence."); Ex.1003, ¶¶26-33.

V.   SUMMARY OF THE '471 PATENT

The '471 patent relates to processing packets at an egress end user node ("EEUN") having a TCP/IP based packet interface. Ex.1001, Abstract, 4:64-5:2; *see also* Ex.1001, 6:59-63 (discussing the OSI protocol stack, as an example). The EEUN includes a decoder, a DMA device, packet buffer memory, and protocol stack layer memory. Ex.1001, 5:3-9. The claimed decoder decodes "the encoded

packet for placing the same in a form that is readable to a particular computer program installed on the EEUN." Ex.1001, 5:10-14; Ex.1003, ¶¶34-35.

According to the '471 patent, conventional EEUN's transferred packets from a MAC layer memory to a CPU main memory, called the "packet buffer memory," "through a serial or parallel memory transfer performed by a CPU." Ex.1001, 5:47-51. Each protocol stack layer writes its respective header into the packet buffer memory, Ex.1001, 5:51-54, and the information is processed "in a sequential manner." Ex.1001, 10:47-48; Ex.1003, ¶36. This prior art is depicted in Fig. 5:
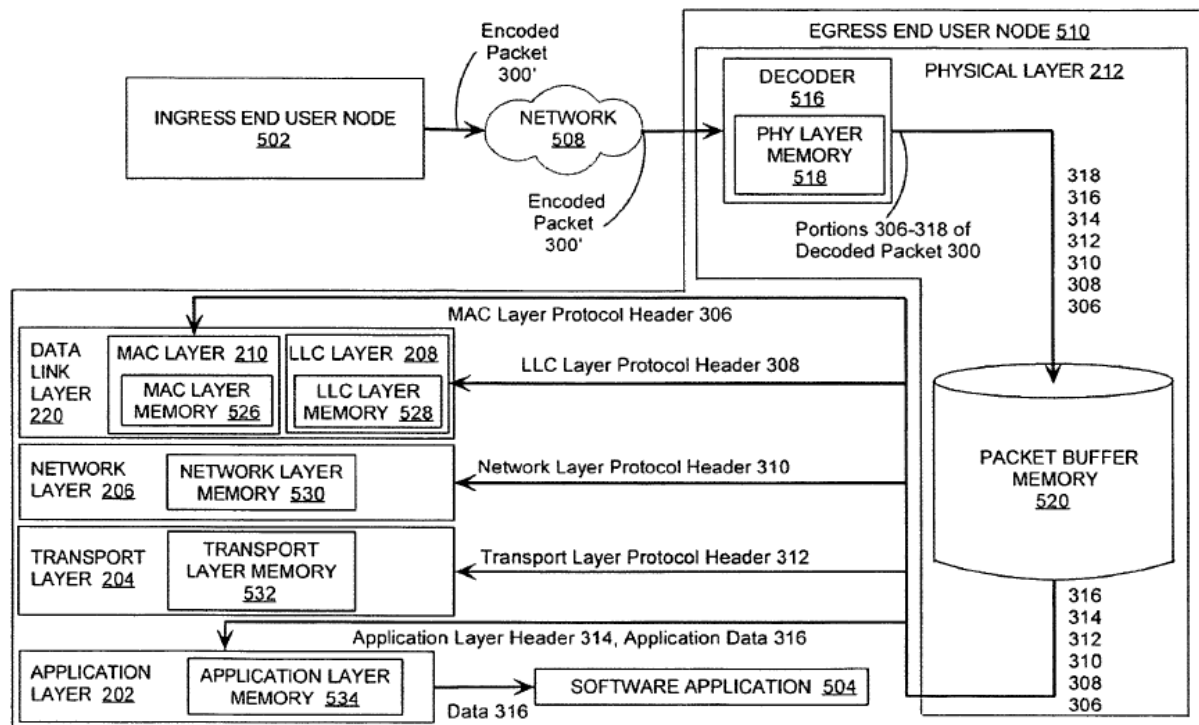
Ex.1001, Fig. 5

While some conventional EEUN's included one or more DMA devices, the '471 patent teaches that the prior art DMA devices "are provided only for a one

time transfer of a packet from the MAC layer memory to a CPU main memory."

Ex.1001, 5:57-60. But a single DMA transfer "does not solve or deal with the

varying latencies and the increased processing consequences present in a high

speed wireless network." Ex.1001, 5:60-63. The '471 patent addresses this alleged

deficiency by using a DMA device to "concurrently writ[e] (1) each of said

plurality of headers to a packet buffer memory and (2) each individual one of said

plurality of headers to a respective protocol stack layer memory[.]" Ex.1001, claim

13 at 12:38-43; Ex.1003, ¶¶37-38. This is depicted in Fig. 1:
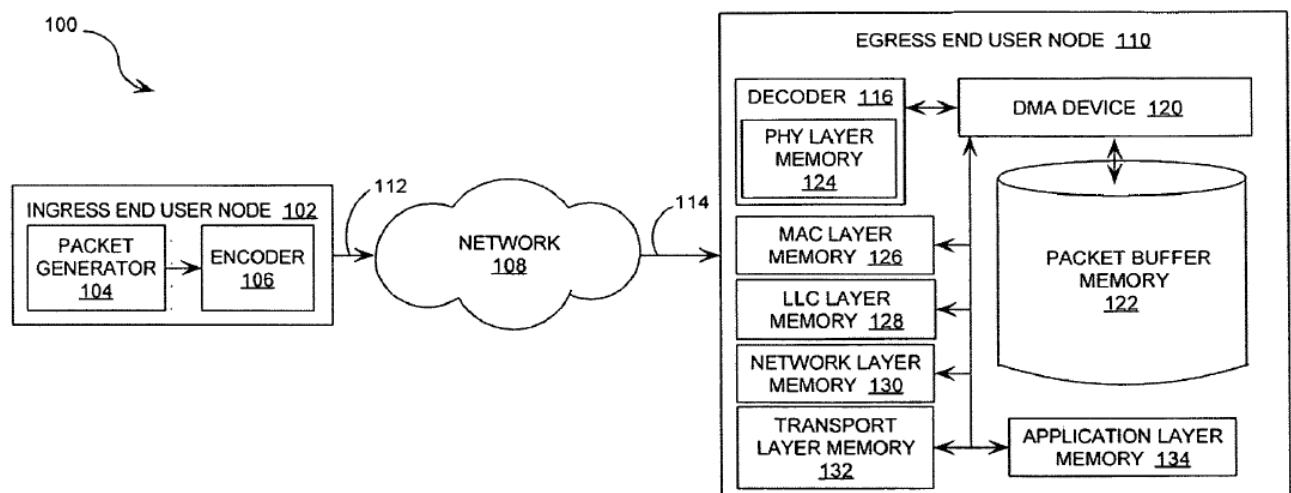
Ex.1001, Fig. 1

As discussed herein, these concepts were well known before the priority date

of the '471 patent. Ex.1003, ¶39.

VI.   EFFECTIVE PRIORITY DATE OF THE '471 PATENT

The '471 patent was filed Sept. 14, 2007, and does not claim priority to any

earlier application.

VII. LEVEL OF ORDINARY SKILL IN THE ART

A Person of Ordinary Skill in The Art ("POSITA") in Sept. 2007 would have had a working knowledge of network communications techniques and high-speed packet processing technologies. Ex.1003, ¶20. A POSITA would have had a bachelor's degree in computer science, computer engineering, electrical engineering, or an equivalent training, and approximately two years of work experience in network communications. Lack of work experience can be remedied by additional education, and vice versa. Ex.1003, ¶20.

VIII. CLAIM CONSTRUCTION

Claim terms in IPR are construed according to their "ordinary and customary meaning" to those of skill in the art. 37 C.F.R. §42.100(b). For the purposes of this proceeding, and the grounds presented herein, Petitioner contends that no claim term requires express construction. Ex.1003, ¶41.

IX. RELIEF REQUESTED AND REASONS FOR THE RELIEF

Petitioner asks the Board to institute trial and cancel the Challenged Claims.

X. STATEMENT OF MATERIAL FACTS

1. The TCP/IP model includes a link layer, a network layer, a transport layer, and an application layer. Ex.1010, [0009]-[0013].

2. The TCP/IP model includes a physical layer as its lowest layer, below the link layer. Ex.1003, ¶27.

3.   The OSI model includes a physical layer, a data link layer, a network layer, a transport layer, a session layer, a presentation layer, and an application layer. Ex.1023, 2.

4.   The application layer of the TCP/IP model corresponds to the combination of the application, presentation, and session layers of the OSI model. Ex.1010, [0010], [0011].

5.   Before the priority date of the '471 patent, network nodes that sent and received packets using a TCP/IP based packet interface were known. Ex.1001, 4:64-5:2; Ex.1005, [0028], [0077].

6.   Before the priority date of the '471 patent, it was typical for packets transmitted using a TCP/IP based packet interface to include multiple encapsulated headers. Ex.1001, 1:40-59; Ex.1009, 19-20.

7.   Before the priority date of the '471 patent, in IEEE 802.3, an Ethernet header was an example of a media access control layer header. Ex.1017, 4:2-4 ("Ethernet MAC headers"); Ex.1019, [0008] ("an Ethernet (MAC) header"); Ex.1018, [0043] ("Ethernet frames…start[] with Media Access Control (MAC) information[.]").

8.   Before the priority date of the '471 patent, in IEEE 802.3, use of a logical link control layer header was required. Ex.1014, 8:61-64; Ex.1015, 2:14-18; Ex.1016, Figs. 5A & 5B, 4:20-28.

9.   Before the priority date of the '471 patent, in IEEE 802.3, it was known that a packet's preamble sequence consisted of 8 bytes (64 bits) where the first 62 bits consisted of alternating 1's and 0's and the last 2 bits marked the end of the preamble sequence and were known as the start of frame delimiter. Ex.1020, 1:30-44.

10.   Before the priority date of the '471 patent, it was conventional to include one or more DMA devices in a network node such that the DMA device was used to transfer a packet from one memory location to another memory location. Ex.1001, 5:57-60; Ex.1005, [0074].

11.   Before the priority date of the '471 patent, storing packet headers in one memory location or buffer and storing the entire packet (header and payload) in another memory location or buffer at a network node was known. Ex.1005, [0081]; Ex.1006, 6:66-7:4.

XI.   IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

A.   Challenged Claims

Lionra Technologies Ltd. has asserted claims 1-3, 5, 8-10, 13-15, 17, and 20-21 against Petitioner. A finding that the Challenged Claims, which include the asserted claims and additional claims, are unpatentable at the PTAB will resolve the parties' dispute in the co-pending litigation and obviate the need for trial, substantially reducing the time and expense of the litigations for all parties.

IPR2024-01281 Petition
Inter Partes Review of U.S. 7,738,471

B.   Grounds

No.	Claims	Basis
Ground 1	1-3, 5-10, 12-15, 17-21	35 U.S.C. §103 (Pre-AIA) over Cornett, Paatela, and Nelson
Ground 2	8	35 U.S.C. §103 (Pre-AIA) over Cornett, Paatela, Nelson, and Russell

U.S. Patent Appl. No. 2006/0072564 (Ex.1005, "**Cornett**"); filed May 26, 2005; published Apr. 6, 2006. Pre-AIA §102(b) prior art.

U.S. Patent Appl. No. 2006/0209840 (Ex.1007, "**Paatela**"); filed Sept. 12, 2005; published Sept. 21, 2006. Pre-AIA §102(e)(1) prior art.

U.S. Patent No. 7,333,489 (Ex.1006, "**Nelson**"); filed Oct. 24, 2000; issued Feb. 19, 2008. Pre-AIA §102(e)(2) prior art.

U.S. Patent No. 6,678,746 (Ex.1025, "**Russell**"); issued Jan. 13, 2004. Pre-AIA §102(b) prior art.

None of these references were before the Examiner during prosecution.

Petitioner also cites to additional prior art (Exs.1008-1020 and 1023-1026) as evidence of a POSITA's background knowledge and to provide contemporaneous context to support Petitioner's assertions regarding what a POSITA would have understood from the prior art in the grounds. This approach has been approved by the Federal Circuit. *Yeda Research v. Mylan Pharm. Inc.*, 906 F.3d 1031, 1041-1042 (Fed. Cir. 2018); 37 C.F.R. §42.104(b).

16

IPR2024-01281 Petition
Inter Partes Review of U.S. 7,738,471

Dr. Mir and this Petition also cite to Request for Comments (RFC) 793. To the extent the Board determines that RFC 793 must qualify as a prior art "printed publication," it does so qualify. *See, e.g.*, Ex.1009, 522 (discussing RFC 793 in 1996, confirming that it was publicly available and known to networking professionals). Indeed, the Board has repeatedly found RFC documents to be "printed publications." *See, e.g., Apple, Inc. v. VirnetX, Inc.*, IPR2017-00337, Paper 31, 46-47 (May 30, 2018) (RFCs are "precisely the type of documents whose main purpose is for public disclosure"); *Riot Games, Inc. v. Paltalk Holdings, Inc.*, IPR2018-00130, Paper 11, 30-33 (May 15, 2018) (RFCs are printed publications).

XII.   GROUND 1: CORNETT-PAATELA-NELSON RENDER OBVIOUS CLAIMS 1-3, 5-10, 12-15, 17-21

A.   Summary of Cornett

Cornett discloses "accelerated" TCP processing where bottlenecks, which might result from using the core processing module of a host processor to perform data movement, are reduced using a "DMA engine." Ex.1005, [0003]-[0005]. "Since the core processing module of a host processor may be bypassed using a DMA engine, slow memory access speeds may be avoided." Ex.1005, [0097]; Ex.1003, ¶45. Cornett's solution is implemented on a network node such as an "intermediate station[], such as, for example, one or more hubs, switches, and/or

17

routers" or an "end station[]." Ex.1005, [0021], [0023]; *see also* Ex.1005, [0038] (discussing the OSI model).

The node has a network interface controller ("NIC") that includes "a physical layer device (PHY) that translates between the analog signals of a communications medium (e.g., a cable or wireless radio) and digital bits." Ex.1005, [0079]. The NIC performs "header splitting" where the NIC "cause[s] storage 704 (e.g., via Direct Memory Access (DMA)) of the packet's header in the page(s) used to store headers and separately store 706 the packet's payload." Ex.1005, [0073]-[0074]. In one embodiment, packet headers 614*a*, 616*a* are stored **both** "in separate buffers" and in a "location in which the payload 614*b*, 616*b* is written." Ex.1005, [0081]; Ex.1003, ¶¶46-47. This is shown in Fig. 10:
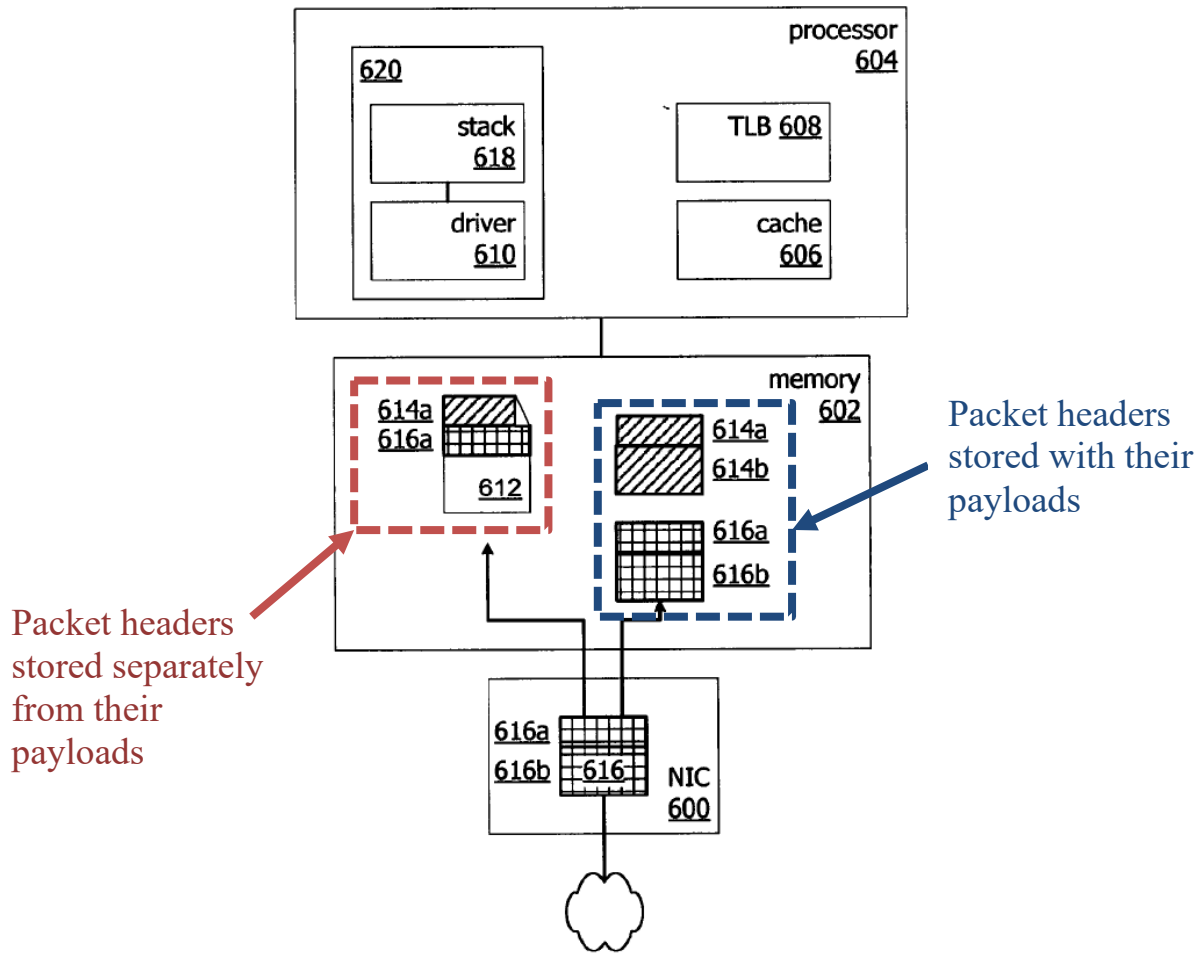
Ex.1005, Fig. 10 (annotated)

Cornett and the '471 patent are both in the field of packet-based network communications and they both are specifically directed to the same topic: accelerating packet header processing. *See* Ex.1001, 1:8-11 ("invention relates to a method and apparatus for high speed protocol header processing[.]"); Ex.1005, [0097] ("TCP/IP processing may be accelerated by using a data movement module, such as a DMA engine, to move data from one buffer to another buffer."); Ex.1003, ¶48. Cornett is also reasonably pertinent to the problem faced by the

19

inventor of the '471 patent: how to accelerate data movement as part of packet processing using a DMA engine. Ex.1001, 6:1-6; Ex.1005, [0097]; Ex.1003, ¶48.

B.   Summary of Nelson

Nelson also accelerates packet processing by making routing decisions "based on the [packet's] header information without reading the packet out of the buffer." Ex.1006, 1:13-20. Nelson notes that routing decisions are ordinarily made by storing a complete ethernet frame in memory and then examining the frame's header. Ex.1006, 1:54-57, 1:60-63. This process, however, requires the frame to be read and stored in multiple locations thereby increasing the time required to forward a frame. Ex.1006, 1:64-67; Ex.1003, ¶49.

Nelson's solution is to "store[] frames of data in the storage element" and "concurrently snoop[] on the frame data to obtain header information" that is "stored in a buffer separate from" the memory that "stores the frames." Ex.1006, 2:13-17. Nelson's separate buffer can be implemented as a FIFO or as "dual port RAM." Ex.1006, 6:8-9; Ex.1003, ¶¶50-51. This is shown in Fig. 5:

Ex.1006, Fig. 5 (annotated)

Nelson and the '471 patent are both in the field of packet-based network communications and they both are specifically directed to the same topic: accelerating packet header processing. *See* Ex.1001, 1:8-11 ("invention relates to a method and apparatus for high speed protocol header processing[.]"); Ex.1006, 3:20-22 (invention "eliminates the need to read a part of the frame out of the FIFO and store this data in another buffer while a routing decision is made."); Ex.1003, ¶52. Nelson is also reasonably pertinent to the problem faced by the inventor of the '471 patent: how to accelerate packet header processing by storing packets and packet headers separately. *See* Ex.1001, 3:9-15 ("writing (1) each of the headers to a packet buffer memory and (2) each individual one of the headers to a respective protocol stack layer memory where it is available for immediate processing within

21

a protocol stack layer."); Ex.1006, 2:3-10 ("wherein the frames of data are stored in a first-in-first-out buffer and wherein the header information is accessed to make routing decisions in order to avoid having to read the frames out of the buffer."); Ex.1003, ¶52.

C.   Summary of Paatela

Paatela seeks to "avoid bogging down" packet routing by a process called "packet transformation." Ex.1007, [0041], [0015]. The process can be implemented at "an intermediary network node between the source and destination nodes" or "at a network node," including a switch. Ex.1007, [0016]-[0017], [0043]; *see also* Ex.1007, [0007] (discussing the OSI model); Ex.1003, ¶53.

Memory 716 on the device implementing the process "stores portions of the packet input 706, and stores these packet portions in memory locations 718-734 as dictated by the input controller 740." Ex.1007, [0076]. In one embodiment, "these packet portions correspond to headers of the various protocol layers associated with the incoming packet." Ex.1007, [0077]. The input controller directs header information to memory 716 and "determines where one networking layer header ends and the next networking layer header begins." Ex.1007, [0077]; Ex.1003, ¶54. This is show in Fig. 7:
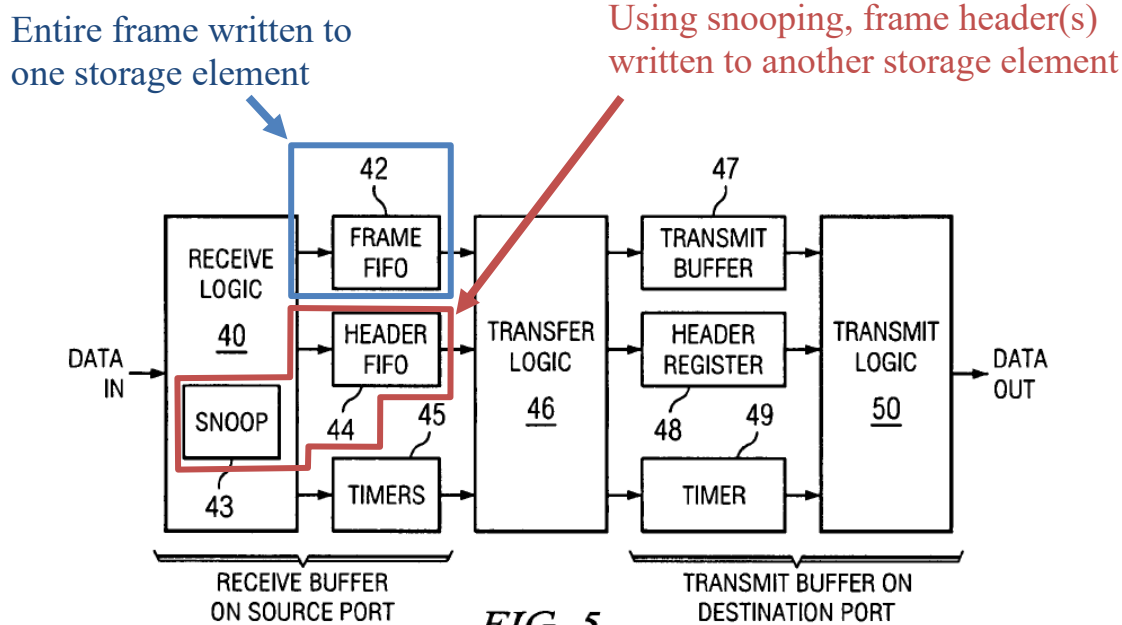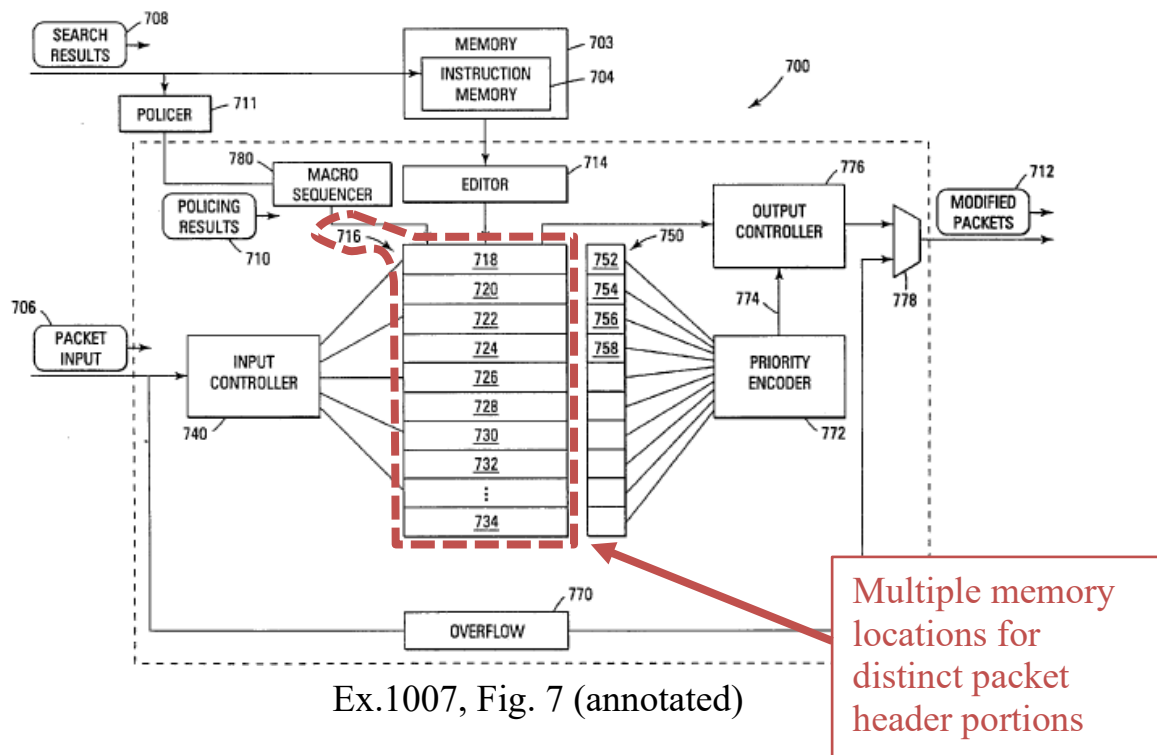
Ex.1007, Fig. 7 (annotated)

Paatela and the '471 patent are both in the field of packet-based network communications and they both are specifically directed to the same topic: accelerated packet header processing. *See* Ex.1001, 1:8-11 ("invention relates to a method and apparatus for high speed protocol header processing[.]"); Ex.1007, [0041] (invention "increase[es] speed and efficiencies of network data throughput."); Ex.1003, ¶55. Paatela is also reasonably pertinent to the problem faced by the inventor of the '471 patent: how to improve processing efficiency by processing headers of various protocol layers individually. *See* Ex.1001, 5:64-6:6 (writing "specific layer header fields directly into: (a) each protocol stack layer's internal memory spaces…[and] reduc[ing] processing latency and thereby increas[ing] bandwidth."); Ex.1007, [0077] ("header information corresponding to

23

OSI networking layers two through four may each correspond to a segment of the memory 716"), [0076] (packet 706 "is stored in the memory 716 in time for the editing instructions from the instruction memory 704 to be processed by the editing processor 714, and for the packet data in the memory 716 to be operated on by the editing processor 714."); Ex.1003, ¶55.

D.   Reasons to Combine Cornett, Paatela, and Nelson

A POSITA would have combined Cornett, Paatela, and Nelson because they all aim to accelerate packet processing. *See* Ex.1005, [0097] (enabling accelerated packet routing decisions by moving data from one processing buffer to another using a DMA engine); Ex.1007, [0076], Abstract, [0079] (accelerating packet processing by moving each one of a packet's encapsulated headers into individual memory segments and processing each header); Ex.1006, 1:13-20, 2:3-10 (accelerating packet routing decisions by eliminating the need to read the entire packet frame from memory, making the decision based on the frame header only); Ex.1003, ¶56. There is motivation to combine where a known technique "has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way." *Intel Corp. v. Qualcomm Inc.*, 21 F.4th 784, 800 (Fed. Cir. 2021) (*citing KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 417 (2007)).

A POSITA would have been motivated to combine Cornett, Paatela, and Nelson to arrive at the claimed invention and the combination would have been within the POSITA's skillset. Ex.1003, ¶57. The combination represents the use of known techniques (writing packet headers into individual memory segments to accelerate processing, per Paatela, writing an entire packet frame into one buffer and the packet headers only into another buffer in parallel to accelerate routing decisions, per Nelson) to improve a similar process (accelerated packet processing technique that stores packet headers separately from an entire packet—both header and payload—using a DMA engine, per Cornett) in the same way to yield predictable results. Ex.1003, ¶57. Such non-benefits-based combinations are permissible. *See Intel Corp. v. PACT XPP Schweiz AG*, 61 F.4th 1373, 1381 (Fed. Cir. 2023) (enough for obviousness to show that features of secondary references are a "suitable option" and there is no need to show that these features are "an 'improvement' in a categorical sense.").

Furthermore, Cornett teaches why accelerated packet processing is beneficial to a network and discloses a technique for doing so by storing packet headers in one buffer and storing the entire packet (header and payload) in another, separate buffer. Ex.1003, ¶58. Cornett does not explicitly disclose how each header of a packet is stored or how each such header is processed. Ex.1003, ¶58. Paatela provides the missing detail by teaching that each individual header is written to a

respective, separate memory segment for processing. Ex.1007, [0077]; *see* Section XII.D.1; Ex.1003, ¶58.

The Cornett-Paatela combination does not explicitly explain the timing of how packet headers are written to Paatela's respective, separate memory segments. Ex.1003, ¶58. Nelson provides the expected, yet missing detail by explaining that packet processing is accelerated, and processing overhead is reduced, by writing the full packet frame into one buffer and the packet headers into another buffer, in parallel. Ex.1006, 2:3-10; *see* Section XII.D.2; Ex.1003, ¶58.

A POSITA would have therefore been motivated to combine Cornett, Paatela, and Nelson. *See Intel*, 61 F.4th at 1381 ("It's enough for Intel to show that there was a known problem of cache coherency in the art, that Bauman's secondary cache helped address that issue, and that combining the teachings of Kabemoto and Bauman wasn't beyond the skill of an ordinary artisan."). A POSITA would also have been motivated to combine the references for the reasons articulated below.

1.   Efficient Data Storage (Cornett-Paatela)

Cornett proposed an accelerated packet processing technique that uses a DMA engine to "move data from one buffer to another buffer." Ex.1005, [0097]. The DMA engine is used in a process called "header splitting" where packet headers are stored in one buffer and the entire packet (header and payload) is

26

stored in another, separate buffer. Ex.1005, [0073], [0081]. Doing so reduces the

number of memory pages "in a way to permit effective prefetching of packet

headers into the processor cache before the protocol stack processes the header."

Ex.1005, [0064]; Ex.1003, ¶59.

A POSITA would have known that Cornett's packets include a plurality of

headers because header encapsulation is a standard part of network

communications. Ex.1003, ¶60. Cornett does not explicitly explain how each of the

headers are stored or processed in its "header splitting" process. Ex.1003, ¶60.

Paatela provides this implementation detail. Ex.1003, ¶61. Paatela teaches that

packets have multiple headers, each of which must be processed to make a routing

decision. Ex.1003, ¶61. To that effect, Paatela teaches separating out each of a

packet's headers and writing each such header to a respective, separate memory

segment for processing. Ex.1007, [0077].

Paatela provides memory 716 that is segmented into "memory locations

718-734," each of which "correspond[s] to headers of the various protocol layers

associated with the incoming packet." Ex.1007, [0077]. Once the headers are

stored in respective, separate memory segments, Paatela makes routing decisions

for each packet and modifies individual headers as needed. For example, if

memory location 718 stores an Ethernet header, "the Ethernet header can be

modified by writing a new Ethernet header into an otherwise unused, interleaved

memory location 720[.]" Ex.1007, [0079]. The new Ethernet header in memory location 720 "effectively replaces the original (now-disregarded) Ethernet header at memory location 718." Ex.1007, [0079]. A POSITA would have appreciated that the combination of Cornett's "header splitting" technique and the use of a DMA engine with Paatela's individual header processing technique, would result in an overall faster system. Ex.1003, ¶61.

The Cornett-Paatela combination is based on the teachings of the references themselves and does not require physical incorporation of elements from Paatela into Cornett. Ex.1003, ¶63. A physical incorporation of Paatela's teachings for how to process individual headers of Cornett's packets would nevertheless be possible for a POSITA and result in the claimed invention. Ex.1003, ¶63. The physical incorporation would use Paatela's input controller and packet classification engine to "determine[] where one networking layer header ends and the next networking layer header begins." Ex.1007, [0077]. It would then use Cornett's DMA engine to move each individual packet header into a respective memory segment, per Paatela. Ex.1005, [0074], [0097]. Thus, it would have been obvious to incorporate Paatela's teachings into Cornett's technique. Ex.1003, ¶63.

A POSITA would have had a reasonable expectation of success in the proposed combination. Ex.1003, ¶62. Cornett teaches how to process packet headers separately from the entire packet (header and payload). Ex.1005, [0081],

[0064]. Paatela merely adds detail about how each header is individually stored in a respective, separate memory segment. Ex.1007, [0077]. Cornett's use of a DMA engine to move data from one buffer to another would simply offload data movement responsibilities from the core processing module of a host processor. Ex.1005, [0097]. A POSITA would have understood Paatela to provide the missing implementation details that Cornett expects a POSITA to already know. Ex.1003, ¶62. The combination would have been within the skillset of the POSITA based on the teachings of the references. Ex.1003, ¶62.

2.   Efficient Data Movement (Cornett-Paatela-Nelson)

Cornett uses a DMA engine to efficiently move packet data from one buffer to another. Ex.1005, [0097]. Doing so "significantly reduce[s] TCP/IP processing overhead that may result from using the core processing module of a host processor." Ex.1005, [0097]. Cornett uses the DMA engine to store packet headers in one buffer and the entire packet (header and payload) in another, separate buffer. Ex.1005, [0081]. The Cornett-Paatela combination would use Cornett's DMA engine to move each of a packet's individual header into a respective memory segment, per Paatela. Ex.1007, [0077]; *see* Section XII.D.1; Ex.1003, ¶64. The timing of how packet headers are written to the respective memory segments is not explicit in the combination. Ex.1003, ¶64.

There are only a few options for how to coordinate the storage of packet headers in multiple memory segments. Ex.1003, ¶65. One option would be to move the data sequentially such that each header is moved to a respective memory segment in a one-at-a-time fashion. Ex.1003, ¶65. Another option would be to move the headers to their respective memory segments in parallel. Ex.1003, ¶65. A POSITA would have understood Cornett to be dedicated to reducing processing overhead and processing time, Ex.1005, [0097], and Paatela to be focused on optimizing packet processing in an environment where time is of the essence. Ex.1007, [0076] (packet 706 "is stored in the memory 716 **in time for** the editing instructions from the instruction memory 704 to be processed by the editing processor 714, and for the packet data in the memory 716 to be operated on by the editing processor 714."); Ex.1003, ¶65. A POSITA would have therefore looked to Nelson to provide the missing detail that is consistent with the Cornett-Paatela combination. Ex.1003, ¶65.

Nelson teaches writing a packet frame (header and payload) into one buffer and the packet headers into another buffer, with the two writing operations occurring in parallel. Ex.1006, 7:1-4. By accomplishing both tasks at the same time, Nelson seeks to accelerate packet processing and reduce processing overhead. Ex.1006, 2:3-10. A POSITA would have appreciated that Nelson

presents an obvious and highly logical design choice consistent with the Cornett-Paatela combination. Ex.1003, ¶65.

A POSITA would have had a reasonable expectation of success in the proposed combination. Ex.1003, ¶66. A POSITA would have appreciated that Paatela stores individual packet headers in respective, separate memory segments to make each packet header available for processing as quickly as possible. Ex.1007, [0076]. Cornett seeks to further accelerate packet processing by using a DMA engine to free up the host processor and move data more efficiently. Ex.1005, [0097]. Nelson similarly seeks to process packets faster by writing headers and frames into separate buffers—in parallel—so that header information is made available for routing decisions without the need (and associated delay) to read the entire frame from the buffer. Ex.1006, 2:3-10. A POSITA would have understood Nelson to disclose a logical design choice for how to further the Cornett-Paatela combination's accelerated packet processing technique. Ex.1003, ¶66. The combination would have been within the skillset of the POSITA based on the teachings of the references. Ex.1003, ¶66.

The Cornett-Paatela-Nelson combination is based on the teachings of the references themselves and does not require physical incorporation of elements from Paatela and Nelson into Cornett. Ex.1003, ¶67. A physical incorporation of the teachings would nevertheless be possible for a POSITA and result in the

31

claimed invention. Ex.1003, ¶67. The physical incorporation would use Paatela's

input controller and packet classification engine to "determine[] where one

networking layer header ends and the next networking layer header begins."

Ex.1007, [0077]. It would then use Cornett's DMA engine to move each individual

packet header into a respective memory segment, per Paatela. Ex.1005, [0074],

[0097]. The move operation would be performed in parallel, per Nelson, to further

accelerate packet processing. Ex.1006, 7:1-4. It would have been obvious to

incorporate Paatela's and Nelson's teachings into Cornett's solution. Ex.1003, ¶67.

E.   Detailed Analysis

1.   Claim 13

[13.0] *An egress end user node (EEUN) of a packet based communications system, comprising:*

To the extent the preamble is limiting, Cornett discloses it. Cornett describes

a network consisting of a "plurality of nodes" that can be "one or more

intermediate stations, such as, for example, one or more hubs, switches, and/or

routers…[and] one or more end stations." Ex.1005, [0021], [0023]; *see also*

Ex.1005, [0004] (discussing TCP/IP packet processing using a TCP/IP stack). The

nodes include a network interface controller ("NIC") and, therefore, disclose the

claimed *egress end user node (EEUN)*. Ex.1005, [0028], [0077]; Ex.1003, ¶¶68-71.

An exemplary node is shown in Fig. 9:
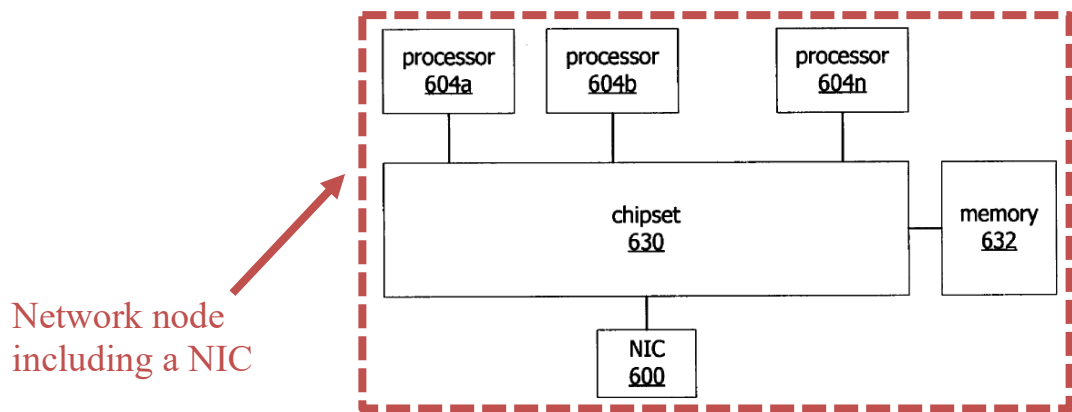
Ex.1005, Fig. 9 (annotated)

This is consistent with the '471 patent that describes the EEUN as a device connected to a network as "an intermediate or destination node" such as a "destination computer system, a routing device…or any other device having a TCP/IP based packet interface." Ex.1001, 4:62-5:2; Ex.1003, ¶72.

Cornett's nodes communicate "via wirelined (e.g., copper wires), or wireless (e.g., radio frequency) means" using "one or more packets." Ex.1005, [0021]; Material Fact No. 5. This discloses *a packet based communications system*. Ex.1003, ¶69.

[13.1] *a decoder configured for decoding a packet having a plurality of headers; and*

Cornett's NIC includes "a physical layer device (PHY)," the claimed *decoder*, in one of its embodiments. Ex.1005, [0079]; Ex.1003, ¶73. The PHY "translates between the analog signals of a communications medium (e.g., a cable or wireless radio) and digital bits." Ex.1005, [0079]. In doing so, the PHY converts

analog-digital representations of a packet as it is transmitted on the communication medium into digital bits, that is, it is *configured for decoding a packet*. Ex.1003, ¶73. The translation process makes a packet readable by digital processing logic in Cornett's "nodes," which are devices that process digital bits. Ex.1003, ¶¶73-74.

The function of Cornett's PHY is consistent with the '471 patent's decoder, which explains that its decoder "performs actions involving the decoding of the encoded packet for placing the same in a form that is readable to a particular computer program installed on the EEUN 110." Ex.1001, 5:10-14; Ex.1003, ¶74.

A POSITA would have known that packets typically include a plurality of headers because a protocol operating on a particular network communication layer adds its own header to the packet, encapsulating the headers from the lower layers. Ex.1008, 1 (TCP fits into "layered hierarchy of protocols[.]"); Ex.1009, 19-20 (explaining header encapsulation in a layered protocol); Ex.1023, 4; *see* Section IV; Material Fact No. 6; Ex.1003, ¶75. As Cornett is concerned with "accelerated TCP…stack processing," Ex.1005, [0003], a POSITA would have understood that Cornett's packets have *a plurality of headers*. Ex.1003, ¶75.

[13.2] *a direct memory access (DMA) device coupled to said decoder and*

Cornett's NIC performs "header splitting" on selected packets. Ex.1005, [0073]. For packets selected for header splitting, the NIC "can cause storage 704 (e.g., via Direct Memory Access (DMA)) of the packet's header[.]" Ex.1005,

[0074]; *see also* Ex.1005, [0097] ("TCP/IP processing may be accelerated by using…a **DMA engine**, to move data from one buffer to another buffer."). Cornett's DMA engine is the claimed *direct memory access (DMA) device*. Ex.1003, ¶76.

Cornett's NIC includes a PHY, the claimed *decoder*, that converts analog-digital representations of a packet as it is transmitted on the communication medium into digital bits. *See* [13.1]; Ex.1005, [0079]; Ex.1003, ¶77. The NIC then causes storage of the digital bits of the packet headers into memory using the DMA engine, the claimed *direct memory access (DMA) device*. Ex.1005, [0074]; Ex.1003, ¶77. A POSITA would have therefore found it obvious that Cornett's DMA engine is *coupled* to the PHY. Ex.1003, ¶77.

[13.3] *[a direct memory access (DMA) device coupled to said decoder and] configured for concurrently writing (1) each of said plurality of headers to a packet buffer memory and (2) each individual one of said plurality of headers to a respective protocol stack layer memory where it is available for immediate processing within a protocol stack layer.*

Cornett's NIC "receives a packet 614 from a network[.]" Ex.1005, [0068]. The NIC determines "whether to perform header splitting" on the packet and, if yes, the NIC "can cause storage 704 (e.g., via Direct Memory Access (DMA)) of the packet's header in the page(s) used to store headers and separately store 706 the packet's payload." Ex.1005, [0073]-[0074]. Therefore, Cornett's DMA engine,

the claimed *direct memory access (DMA) device*, is *configured for…writing* packet headers. Ex.1003, ¶78.

> a.   *direct memory access (DMA) device…writing…each of said plurality of headers to a packet buffer memory*

Cornett's Fig. 10 embodiment receives packet 616 with header 616*a* (**green**) and payload 616*b* (**purple**). Ex.1003, ¶79. Cornett's NIC then "store[s] the header 614*a*, 616*a* and payload 614*b*, 616*b* in separate buffers, and additionally store[s] the header 614*a*, 616*a* to a location in which the payload 614*b*, 616*b* is written." Ex.1005, [0081]; *see also* Ex.1005, [0068]. A POSITA would have understood Cornett's Fig. 10 embodiment to teach that Cornett's NIC causes storage of the packet headers 614*a* and 616*a* in buffer 612 (**red** outline) and the entire packet (header and payload, 614*a* and *b* and 616*a* and *b*) in another, separate buffer (**blue** outline). Ex.1003, ¶¶79-80. This is shown in Fig. 10:
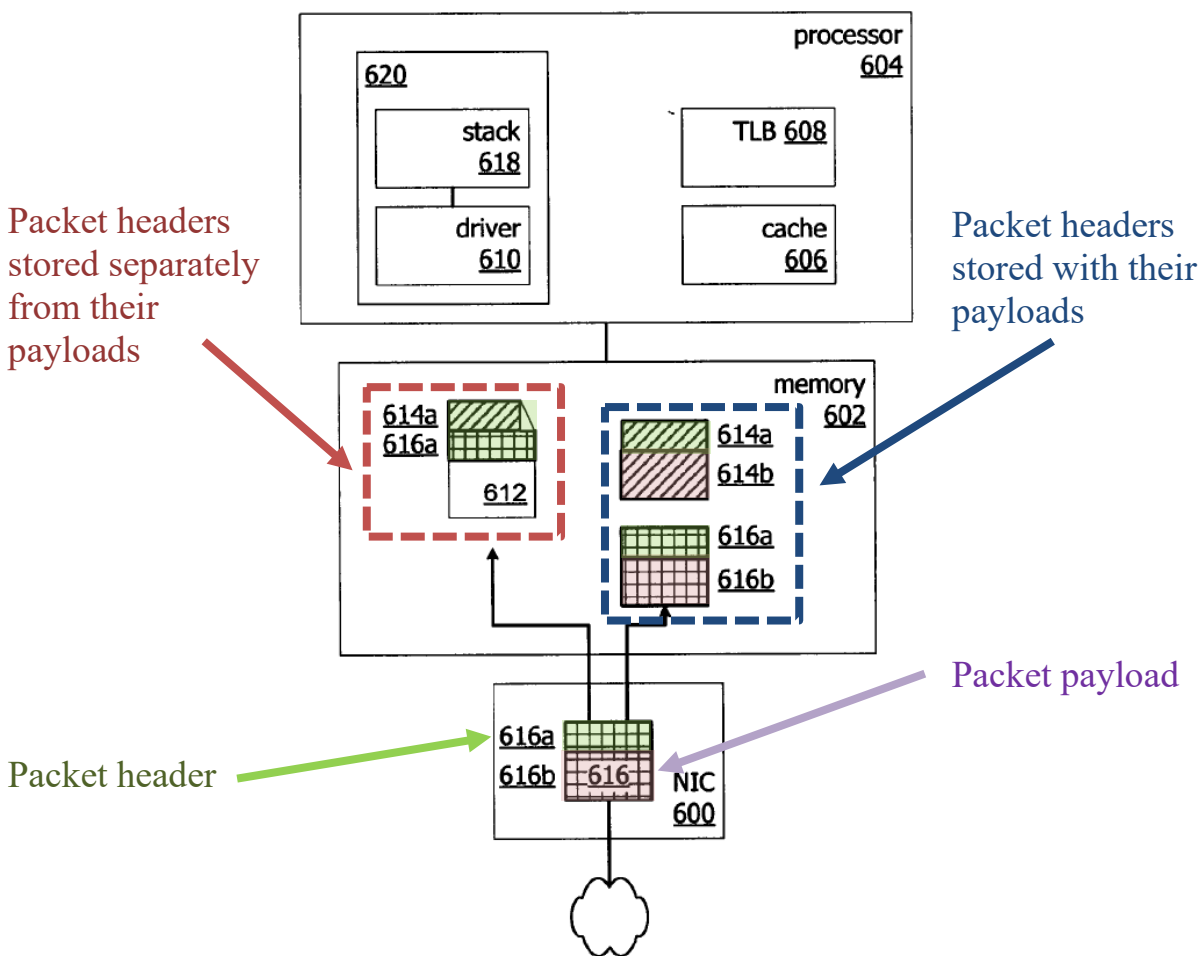
Ex.1005, Fig. 10 (annotated)

The buffer where the entire packet (header and payload) is stored is the claimed *packet buffer memory* because a *plurality of headers* is stored there. Ex.1003, ¶80.

A POSITA would have understood, or at least it would have been obvious, that Cornett's Figs. 9 and 10 embodiments use Cornett's DMA engine to write packet headers and the entire packets (header and payload) to memory. Ex.1003, ¶81. Cornett explains that its invention may significantly reduce TCP/IP processing

37

overhead by using "a data movement module, such as a DMA engine, to move data from one buffer to another buffer." Ex.1005, [0097]. This would express to a POSITA that Cornett expects to use the DMA engine throughout its invention. Ex.1003, ¶81. Cornett's Fig. 7 embodiment also explicitly teaches using the DMA engine to write headers and payloads to memory. Ex.1003, ¶81. Cornett states that Fig. 9 "illustrates a sample computer architecture that can implement the techniques described above," which would include the Fig. 7 embodiment. Ex.1005, [0077]. And Figs. 7, 9, 10 include NIC 600. Ex.1003, ¶81. A POSITA would have thus understood that Fig. 7's DMA engine would operate in the same manner in the Figs. 9 and 10 embodiments. Ex.1003, ¶81. As Cornett's NIC causes storage of headers into memory using its DMA engine, Ex.1005, [0074]; *see* [13.2], a POSITA would have understood that Cornett's DMA engine is *configured for [] writing* packet headers into a *packet buffer memory*. Ex.1003, ¶82.

> b. *direct memory access (DMA) device…writing… each individual one of said plurality of headers to a respective protocol stack layer memory*

Cornett teaches storing packet headers in a buffer but does not explicitly disclose how each header is stored or processed. Ex.1003, ¶83. This is disclosed by Paatela. Paatela teaches routing packets using a process called "packet transformation" where packets are "temporarily stored, and acted upon through processing of protocol-dependent instructions." Ex.1007, Abstract. Paatela's

38

system 700 receives packets/frames as input 706. Ex.1007, [0060]. Input controller 740 determines "where one networking layer header ends and the next networking layer header begins" and uses this information to direct header information to memory 716. Ex.1007, [0077]; Ex.1003, ¶84.

Memory 716 is "configured and partitioned such that all header information and data is stored within the memory 716." Ex.1007, [0085]. Memory 716 (**red** outline) "stores portions of the packet input 706…in memory locations 718-734 [(**green**)] as dictated by the input controller 740." Ex.1007, [0076]. "[T]hese packet portions correspond to headers of the various protocol layers associated with the incoming packet." Ex.1007, [0077]. For example:

> **header information corresponding to OSI networking layers two through four may each correspond to a segment of the memory 716**, such that a **segment of memory 716 is allocated to store a layer-2 header** (e.g., a PPP header), **a segment corresponding to a layer-2.5 header** (e.g., an MPLS label stack), **a segment corresponding to a layer-3 header** (e.g., an IP header), and **a segment corresponding to a layer-4 header** (e.g., a TCP header).

Ex.1007, [0077]; Ex.1003, ¶84. This is shown in Fig. 7:
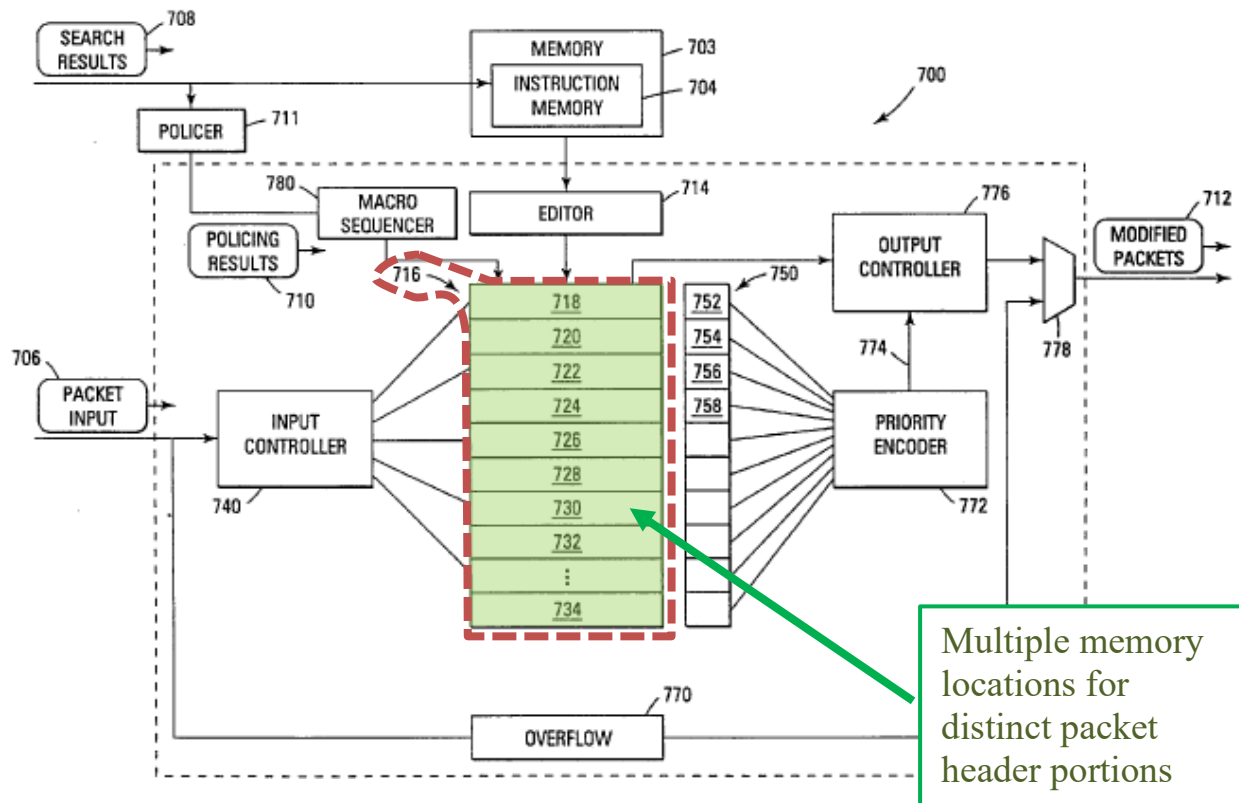
Ex.1007, Fig. 7 (annotated)

Paatela therefore discloses that *each individual one* of a packet's *plurality of headers* is *written* into a separate segment of memory where each segment is allocated to store a particular OSI layer header. Ex.1003, ¶85. Each one of these segments of memory corresponds to a *respective protocol stack layer memory* because each segment stores an individual layer packet header. Ex.1003, ¶85.
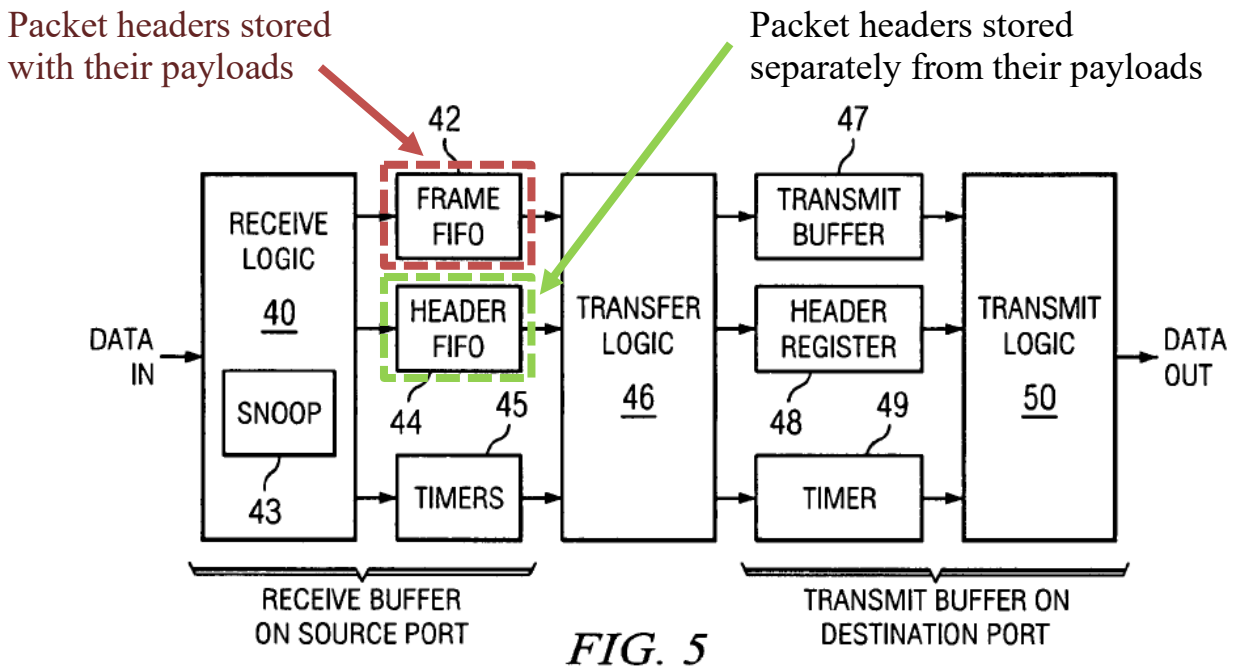
A POSITA would have been motivated to combine Cornett and Paatela because Paatela provides implementation detail that is missing from Cornett. *See* Section XII.D; Ex.1003, ¶86. A POSITA would have understood that Cornett's packets include a plurality of headers. *See* [13.1]; Ex.1003, ¶86. Paatela provides

details about storing and processing multiple layers of packet headers by describing how each individual header, e.g., layer 2, 2.5, 3, and 4, is written to a respective, separate segment of memory for processing. Ex.1003, ¶86. In the combination, Cornett's NIC would cause each individual packet header to be stored into a respective, separate segment of memory for processing, per Paatela. Ex.1007, [0077]; Ex.1003, ¶86. As explained above, Cornett's NIC uses the DMA engine to store headers in memory. Ex.1005, [0074]; *see* [13.2]; Ex.1003, ¶86. It would therefore have been obvious to a POSITA for Cornett's DMA engine to be *configured for [] writing…each individual one of said plurality of headers to a respective protocol stack layer memory*. Ex.1003, ¶86.

Both Cornett and Paatela discuss the OSI model. Ex.1005, [0038]; Ex.1007, [0007]. Cornett discusses TCP/IP packet processing, Ex.1005, [0004], and Paatela processes packet headers that "correspond to headers of the various protocol layers associated with the incoming packet." Ex.1007, [0077]. And both Cornett and Paatela process Ethernet packets. Ex.1005, [0079] (performing "Ethernet frame handling"); Ex.1007, [0079] ("memory location 718 stores an Ethernet header[.]"). A POSITA would have therefore understood that both Cornett and Paatela can operate on TCP/IP packets. Ex.1003, ¶87.

  c. *direct memory access (DMA) device…concurrently writing*

The Cornett-Paatela combination does not explicitly explain the timing of how Cornett's packet headers are written to Paatela's respective, separate segments of memory. This timing is explained in Nelson. Ex.1003, ¶88. Nelson discloses accelerated packet processing "based on the [packet's] header information without reading the packet out of the buffer." Ex.1006, 1:13-20. Packet frames are received by receive logic 40. Ex.1006, 6:62-63. Receive logic 40 "snoop[s] on the received data" and "cop[ies] the header of each frame to header FIFO 44." Ex.1006, 6:66-7:4. The copying of a frame's header to header FIFO 44 "occurs **in parallel** with the storage of the corresponding frame in frame FIFO 42." Ex.1006, 6:66-7:4. According to Nelson, "[d]ual port RAM can be used to store the header information" instead of a FIFO. Ex.1006, 6:8-9. A POSITA would have understood Nelson to teach that the packet's header is *written* (1) to a frame FIFO 42 (header and payload) (**red** outline) and (2) to the header FIFO 44 (header only) (**green** outline). This process is performed in parallel, i.e., *concurrently*. Ex.1003, ¶89. This is shown in Fig. 5:

Ex.1006, Fig. 5 (annotated)

A POSITA would have been motivated to combine Cornett, Paatela, and

Nelson. *See* Section XII.D; Ex.1003, ¶90. Cornett's technique reduces processing

overhead and time. Ex.1005, [0097]. Paatela also optimizes packet processing in a

solution where time is of the essence. Ex.1007, [0076] (packet "is stored in the

memory 716 in time for the editing instructions from the instruction memory 704

to be processed by the editing processor 714, and for the packet data in the

memory 716 to be operated on by the editing processor 714."). Nelson similarly

accelerates packet processing by writing an entire packet to one buffer and just the

packet headers into another buffer in parallel because this allows Nelson to make

routing decisions without reading the full frame out of the frame buffer. Ex.1006,

2:3-10. A POSITA would thus have been motivated to combine Cornett, Paatela,

and Nelson because Nelson's parallel writing would further accelerate packet

processing, which is Cornett's and Paatela's goal as well. Ex.1003, ¶90.

In the combination, Cornett's NIC would cause its DMA engine to store the

packet (header and payload) in a first buffer, the claimed *packet buffer memory*,

and each individual packet header in a separate segment of memory, the claimed

respective protocol stack layer memory, for processing, per Paatela. Ex.1005,

[0081]; Ex.1007, [0077]; Ex.1003, ¶91. Cornett's NIC would cause the DMA

engine to perform both storage operations in parallel, i.e., concurrently, per Nelson.

Ex.1006, 6:66-7:4; Ex.1003, ¶91.

> d.   *available for immediate processing within a protocol stack layer*

Paatela explains that packet 706 "is stored in the memory 716 **in time for**

the editing instructions from the instruction memory 704 to be processed by the

editing processor 714, and for the packet data in the memory 716 **to be operated**

on by the editing processor 714." Ex.1007, [0076]. A POSITA would have

understood that Paatela's focus on completing its storage step "in time" for

processing teaches that time is of the essence. Ex.1003, ¶92. It would therefore

have been obvious to a POSITA that packet headers are written to Paatela's

respective, separate segments of memory such that they are *available for*

immediate processing. Ex.1003, ¶92. This is consistent with Nelson making header

information stored in its header buffer available for immediate routing decisions

without reading the entire frame out of a frame buffer. Ex.1006, 2:3-10. Indeed,

Nelson's use of "dual port RAM" to store headers, Ex.1006, 6:8-9, would facilitate

immediate processing because data written into dual port RAM can be read out of

the RAM at the same time. It would therefore have been obvious to a POSITA that,

in the combination, a packet's headers would be *available for immediate*

processing even before the payload of the frame (which is generally received after

the headers) continues to be written into the frame buffer. Ex.1003, ¶92.

Paatela further explains that its editing processor 714 operates on each stored

packet header "to perform packet modifications and provide packet steering

information" depending on the particular networking protocol. Ex.1007, [0063]. As

each header would be processed by the relevant protocol operating within its

respective layer, it would therefore have been obvious to a POSITA that **each** of a

packet's headers would be *available for…processing within a protocol stack layer*.

Ex.1007, [0063] ("The editor instructions are executed to perform packet

modifications and provide packet steering information.…Such special purpose

instructions may be particularly useful to perform certain networking-specific tasks

that depend on the particular networking protocol."); Ex.1003, ¶93.

2.   Claim 14

[14.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a media access control layer protocol header and said respective protocol stack layer memory is a media access control layer memory.*

First, a POSITA would have understood that both Cornett and Paatela can operate on TCP/IP packets. *See* [13.3]. Both Cornett and Paatela handle Ethernet packets. Ex.1005, [0079] (performing "Ethernet frame handling"); Ex.1007, [0079] ("memory location 718 stores an Ethernet header[.]"), [0113] ("layer-2 Ethernet protocol header"). A POSITA would have known that an Ethernet header is an example of a *media access control layer protocol header*. *See, e.g.,* Ex.1017, 4:2-4 ("Ethernet MAC headers"); Ex.1019, [0008] ("an Ethernet (MAC) header"); Ex.1018, [0043] ("Ethernet frames…start[] with Media Access Control (MAC) information[.]"); Material Fact No. 7; Ex.1003, ¶94. It would have therefore been obvious to a POSITA that Cornett and Paatela's packets would include a *media access control layer protocol header*. Ex.1003, ¶94.

Second, Paatela teaches that a packet frame's OSI layer 2 header is stored in "a segment of memory 716 [that] is allocated to store a layer-2 header (e.g., a PPP header)[.]" Ex.1007, [0077]. The portion of memory segment 716 that stores this information is therefore the claimed *media access control layer memory* at least because an OSI layer 2 header includes the MAC header. *See* Ex.1023, 3 (OSI layer 2 is the data link layer); Ex.1010, [0048] ("Within the link layer 240, there is

46

a media access control (MAC) unit 242, a device 243, a transmit queue 244, a

receive queue 245 and a MAC header 246."); Ex.1003, ¶95. Paatela also teaches

storing an "Ethernet header" in a specific "memory location 718" separate from

other protocol headers. Ex.1007, [0079]; *see also* Ex.1007, [0113], [0120] (giving

examples of storing an "Ethernet header" in a separate "memory location"). The

separate memory location 718 alternatively corresponds to the claimed *media*

access control layer memory. Ex.1003, ¶95. It would have been obvious to a

POSITA to store a specific protocol layer header in a corresponding layer of

memory. *See* [13.3]; Ex.1003, ¶95.

3.   Claim 15

[15.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a logic link control layer protocol header and said respective protocol stack layer memory is a logic link control layer memory.*

First, a POSITA would have understood that both Cornett and Paatela can

operate on TCP/IP packets. *See* [13.3]. Both Cornett and Paatela handle Ethernet

packets. Ex.1005, [0079] (performing "Ethernet frame handling"); Ex.1007, [0079]

("memory location 718 stores an Ethernet header[.]"), [0113] ("layer-2 Ethernet

protocol header"). A POSITA would have known that an Ethernet frame header

may optionally include a *logic link control layer protocol header*. Ex.1003, ¶96.

"Ethernet" commonly refers to both the "Ethernet" networking standard (also

known as DIX Ethernet) and to the IEEE 802.3 standard. Ex.1014, 8:42-51;

Ex.1015, 2:4-7 ("Ethernet, both according to the Dec-Intel-Xerox (DIX) and IEEE 802.3 standards"). The IEEE "802.3 Ethernet" standard "require[s] use of a logical link control (LLC) header." Ex.1015, 2:14-18; *see also* Ex.1014, 8:61-64 ("802.3 MAC frame has an IEEE 802.2 logical link control (LLC) header following the MAC header and preceding the OSI network layer protocol header, if one exists."); Ex.1016, Figs. 5A & 5B, 4:20-28; Material Fact No. 8. It would therefore have been obvious to a POSITA for the "Ethernet" frames of Cornett and Paatela to include a *logic link control layer protocol header*. Ex.1003, ¶96.

Second, Paatela teaches that a packet frame's OSI layer 2 header is stored in "a segment of memory 716 [that] is allocated to store a layer-2 header (e.g., a PPP header)[.]" Ex.1007, [0077]. The portion of memory segment 716 that stores this information is therefore the claimed *logic link control layer memory* at least because, in IEEE 802.3 Ethernet, a layer 2 header includes the LLC header. *See* Ex.1023, 3 (OSI layer 2 is the data link layer); Ex.1011, 3:24-26 (data-link layer includes two sub-layers: media access control (MAC) sublayer and a logical link control (LLC) sublayer); Ex.1003, ¶97. Paatela also teaches storing an "Ethernet header" in a specific "memory location 718" separate from other protocol headers. Ex.1007, [0079]; *see also* Ex.1007, [0113], [0120] (giving examples of storing an "Ethernet header" in a separate "memory location"). The separate memory location 718 alternatively corresponds to the claimed *logic link control layer memory*.

48

Ex.1003, ¶97. It would have been obvious to a POSITA to store a specific protocol

layer header in a corresponding layer of memory. *See* [13.3], [14.0]; Ex.1003, ¶97.

4.   Claim 17

[17.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a network layer protocol header and said respective protocol stack layer memory is a network layer memory.*

First, the TCP/IP model includes a network layer that a POSITA would

have known "corresponds to the network layer of the OSI model." Ex.1010,

[0044], [0012]. A POSITA would have known that the network layer includes the

same information in both the TCP/IP and OSI models. Ex.1003, ¶98. Paatela also

teaches that an IP header is an example of a header at the network layer (i.e., layer

3) in the OSI model. Ex.1007, [0077] ("header information corresponding to OSI

networking layers two through four may each correspond to a segment of the

memory 716, such that a segment of memory 716 is allocated to store…a segment

corresponding to a layer-3 header (e.g., an IP header)[.]"). A POSITA would have

known, or it would have been obvious, that the IP header is found in the network

layer and is a *network layer protocol header*. Ex.1003, ¶99.

Second, Paatela teaches that a packet frame's OSI layer 3 header is stored in

"a segment of memory 716 [that] is allocated to store a…layer-3 header (e.g., an IP

header)[.]" Ex.1007, [0077]. Paatela provides other examples where an IP packet

header is written to an IP header memory. Ex.1007, [0113], [0120], [0125]. A

POSITA would have understood that both Cornett and Paatela can operate on TCP/IP packets. *See* [13.3]. The portion of memory segment 716 that stores the IP header (which is an example of a layer 3 header) is the claimed *network layer memory*. *See* [13.3]; Ex.1003, ¶100.

5.   Claim 18

[18.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a transport layer protocol header and said respective protocol stack layer memory is a transport layer memory.*

First, the TCP/IP model includes a transport layer that a POSITA would have known "corresponds to the transport layer of the OSI model." Ex.1010, [0044], [0011]. A POSITA would have known that the transport layer includes the same information in both the TCP/IP and OSI models. Ex.1003, ¶101. Paatela also teaches that a TCP header is an example header at the transport layer in the OSI model. Ex.1007, [0077] ("header information corresponding to OSI networking layers two through four may each correspond to a segment of the memory 716, such that a segment of memory 716 is allocated to store…a segment corresponding to a layer-4 header (e.g., a TCP header)[.]"). A POSITA would have known, or it would have been obvious, that the TCP header is found in the transport layer and is a *transport layer protocol header*. *See* Ex.1001, 7:60-62 ("transport protocols include a transmission control protocol (TCP)"); Ex.1003, ¶102.

Second, Paatela teaches that a packet frame's OSI layer 4 header is stored in "a segment of memory 716 [that] is allocated to store a…layer-4 header (e.g., a TCP header)[.]" Ex.1007, [0077]. Paatela provides other examples where a TCP packet header is written to a TCP header memory. Ex.1007, [0125], [0130]. The User Datagram Protocol (UDP) was another well-known transport protocol, and Paatela similarly describes storing a UDP header in a separate UDP header memory. Ex.1007, [0113], [0120]. POSITA would have understood that both Cornett and Paatela can operate on TCP/IP packets. *See* [13.3]. The portion of memory segment 716 that stores the TCP or UDP header (which are examples of a layer 4 header) is the claimed *transport layer memory*. *See* [13.3]; Ex.1003, ¶103.

6.   Claim 19

[19.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is an application layer header and said respective protocol stack layer memory is an application layer memory.*

First, the TCP/IP model includes an application layer that a POSITA would have known "corresponds to the combination of the application, presentation and session layers of the OSI model." Ex.1010, [0010], [0011]; Material Fact No. 4. A POSITA would have known that the TCP/IP application layer includes the same information as the combination of OSI's application, presentation, and session layers. Ex.1003, ¶104. It would have been obvious to a POSITA for the application layer to include an application header, which is an *application layer header*. *See,*

51

e.g., Ex.1013, 2:43-48 ("application layer attaches an application header to the payload data[.]"); Ex.1003, ¶105.

Second, Paatela teaches that packets include portions that correspond to "various protocol layers" and packet header information corresponding to OSI networking layers "may each correspond to a segment of the memory 716." Ex.1007, [0077]. Paatela addresses storing layer 2-4 headers in memory 716 and places no limits on which headers it can process. Ex.1003, ¶106. It would therefore have been obvious to a POSITA, based on Paatela's teachings, to store and process the application layer header, when present, in a separate portion of memory segment 716, which corresponds to the claimed *application layer memory*. *See* [13.3]; Ex.1003, ¶106.

7.   Claim 20

[20.0] *The EEUN according to claim 13, wherein said decoder is further configured for communicating a portion of said packet to said DMA device subsequent to decoding said packet.*

Cornett's NIC receives packets and, for those packets selected for header splitting, the NIC "cause[s] storage 704 (e.g., via Direct Memory Access (DMA)) of the packet's header in the page(s) used to store headers and separately store 706 the packet's payload." Ex.1005, [0068], [0073], [0074]; *see* [13.2]; Ex.1003, ¶107. The NIC includes a "physical layer device (PHY)," the claimed *decoder*, that is "communicatively coupled to a media access controller (MAC) (e.g., via a FIFO)

that performs 'layer 2' operations (e.g., Ethernet frame handling)." Ex.1005, [0079]; *see* [13.1]; Ex.1003, ¶108.

A POSITA would have understood Cornett's disclosure as follows: (1) Cornett's NIC receives signals representing packets from the communications medium (Ex.1005, [0079]); and (2) the NIC's PHY "translates," i.e., converts, the analog-digital representations of packets into digital bits (Ex.1005, [0079]). Ex.1003, ¶109. It would have been obvious to a POSITA that the NIC's PHY translates the representations before the digital bits of the packets are transmitted for storage by means of Cornett's NIC using Cornett's DMA engine. Ex.1005, [0074]; Ex.1003, ¶109. This is consistent with Cornett's description of memory 602 as a "collection of physical pages of contiguous memory addresses" and a POSITA's knowledge that physical memory is a form of digital memory. Ex.1005, [0065]; Ex.1003, ¶109. A POSITA would have known that it is generally not feasible to directly store an analog signal. Ex.1003, ¶109. It would have been obvious to translate the signal into digital data and then store that digital data in digital memory. Ex.1003, ¶109. Cornett thus renders obvious that its PHY, the claimed *decoder*, is *further configured* to *communicat[e] a portion of said packet to said DMA device subsequent to decoding said packet*. Ex.1003, ¶109.

8.   Claim 21

[21.0] *The EEUN according to claim 20, wherein said portion is absent of a preamble and a physical layer header.*

First, IEEE 802.3 defines the structure of Ethernet frames, that is, packets. *See* [15.0]; Ex.1003, ¶110. A POSITA would have further known that a node starts the transmission of a packet by "sending an 8 byte (64 bit) preamble sequence." Ex.1020, 1:30-44; Ex.1003, ¶110. The first 62 bits of this sequence are used to synchronize clocks on the sender and receiver. Ex.1020, 1:30-44. The last 2 bits of the sequence are "known as the start of frame delimiter[.]" Ex.1020, 1:30-44; Material Fact No. 9; Ex.1003, ¶110. A POSITA would have known that (1) a frame's physical layer is the lowest layer in both TCP/IP and OSI models and (2) the start of frame delimiter is followed by the data communicated at the next-higher layer (e.g., a layer 2 header followed by a layer 2 payload). *See* Section IV; Ex.1009, 280-281; Material Fact Nos. 2, 3; Ex.1003, ¶111. It would therefore have been obvious to a POSITA that the first 62 bits of the preamble sequence are the claimed *preamble* and the last 2 bits, known as the start of frame delimiter, are the claimed *physical layer header*. Ex.1003, ¶111.

Second, Cornett's PHY, the claimed *decoder*, may be "communicatively coupled to a media access controller (MAC) (e.g., via a FIFO) that performs 'layer 2' operations (e.g., Ethernet frame handling)." Ex.1005, [0079]. And it was well-

known that an Ethernet receive interface "starts collecting the bits into bytes for processing by the MAC layer" only after the last two bits of the preamble—start of frame delimiter—is received. Ex.1020, 1:44-46; Ex.1003, ¶112.

A POSITA would have thus understood that Cornett's NIC receives signals representing packets from the communications medium (Ex.1005, [0079]) and the NIC's PHY "translates," i.e., converts, the analog-digital representations of packets into digital bits (Ex.1005, [0079]). *See* [20.0]; Ex.1003, ¶112. A POSITA would have understood that Cornett's PHY translates the entire packet **and then** passes the information **that follows** the preamble and start of frame delimiter for processing. Ex.1020, 1:44-46; Ex.1003, ¶112. Cornett thus renders obvious that Cornett NIC's PHY, the claimed *decoder*, transmits packets (header and payload)—*absent of* the *preamble* and *physical layer header*—using Cornett's DMA engine for storage. Ex.1005, [0074]; Ex.1003, ¶112.

9.   Claim 1

[1.0] *A method for processing a packet at an egress end user node, comprising:*

To the extent the preamble is limiting, Cornett discloses it. Cornett's nodes include a NIC and, therefore, disclose the claimed *egress end user node (EEUN)*. *See* [13.0]; Ex.1003, ¶¶113-114. Cornett's NIC performs a process called "header splitting," which is the claimed *method for processing a packet*. Ex.1005, [0073]; Ex.1003, ¶114.

55

[1.1] *decoding a packet having a plurality of headers; and*

Limitation [1.1] is substantively identical to limitation [13.1] and is obvious for the same reasons. Ex.1003, ¶115.

[1.2] *subsequent to said decoding step, concurrently writing (1) each of said plurality of headers to a packet buffer memory and (2) each individual one of said plurality of headers to a respective protocol stack layer memory where it is available for immediate processing within a protocol stack layer.*

Limitation [1.2] includes elements from limitations [13.3] and [20.0] with minor differences. Unlike limitation [13.3], limitation [1.2] does not recite a DMA device and, instead, recites "concurrently writing" without being tied to a DMA device of a decoder. Limitation [1.2] is thus obvious for the same reasons discussed in connection with limitation [13.3] at least because this limitation is broader. Also, limitation [1.2] is broader than limitation [20.0] because limitation [20.0] is limited to the use of a DMA device, whereas limitation [1.2] only recites "concurrently writing" information "subsequent to said decoding step." Limitation [1.2] is thus obvious for the same reasons discussed in connection with limitation [20.0]. Ex.1003, ¶116.

10.   Claims 2, 3, 5-7, 9-10

Claims 2, 3, 5-7, 9-10 are substantively identical to claims 14, 15, 17-21, respectively, and are obvious for the same reasons. Ex.1003, ¶¶117-121, 125-126.

[2.0] *The method according to claim 1, wherein at least one of said plurality of headers is a media access control layer protocol header and said respective protocol stack layer memory is a media access control layer memory.*	[14.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a media access control layer protocol header and said respective protocol stack layer memory is a media access control layer memory.*
[3.0] *The method according to claim 1, wherein at least one of said plurality of headers is a logic link control layer protocol header and said respective protocol stack layer memory is a logic link control layer memory.*	[15.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a logic link control layer protocol header and said respective protocol stack layer memory is a logic link control layer memory.*
[5.0] *The method according to claim 1, wherein at least one of said plurality of headers is a network layer protocol header and said respective protocol stack layer memory is a network layer memory.*	[17.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a network layer protocol header and said respective protocol stack layer memory is a network layer memory.*
[6.0] *The method according to claim 1, wherein at least one of said plurality of headers is a transport layer protocol header and said respective protocol stack layer memory is a transport layer memory.*	[18.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is a transport layer protocol header and said respective protocol stack layer memory is a transport layer memory.*
[7.0] *The method according to claim 1, wherein at least one of said plurality of headers is an application layer header and said respective protocol stack layer memory is an application layer memory.*	[19.0] *The EEUN according to claim 13, wherein at least one of said plurality of headers is an application layer header and said respective protocol stack layer memory is an application layer memory.*

57

[9.0] *The method according to claim 1, further comprising communicating a portion of said packet to a direct memory access (DMA) device subsequent to said decoding step and prior to said concurrently writing step.*	[20.0] *The EEUN according to claim 13, wherein said decoder is further configured for communicating a portion of said packet to said DMA device subsequent to decoding said packet.*
[10.0] *The method according to claim 9, wherein said portion is absent of a preamble and a physical layer header.*	[21.0] *The EEUN according to claim 20, wherein said portion is absent of a preamble and a physical layer header.*

11. Claim 8

[8.0] ***The method according to claim 1, further comprising concurrently processing said plurality of headers in each of a plurality of protocol stack layers.***

A POSITA would have understood that time is of the essence in Paatela. Ex.1007, [0076] (packet 706 "is stored in the memory 716 **in time for the editing instructions** from the instruction memory 704 to be processed by the editing processor 714, and for the **packet data in the memory 716 to be operated on by the editing processor 714**."); *see* [13.3]; Ex.1003, ¶122. It would therefore have been obvious to a POSITA for headers to be written to memory in such a way that they become available for immediate processing. Ex.1003, ¶122.

In Paatela, once the packet's headers are stored in segments of memory, and individual headers are modified as needed, Paatela reassembles the packet with the modified headers "in the proper order." Ex.1007, [0079]; Ex.1003, ¶123. In Paatela's Ethernet header example, the modified Ethernet header in memory

location 720 "effectively replaces the original (now-disregarded) Ethernet header

at memory location 718." Ex.1007, [0079]. As processing should not be impacted

by delayed processing of a header, concurrent processing would have made sense

to a POSITA. Ex.1003, ¶123. It would therefore have been obvious to a POSITA

that Paatela's *processing said plurality of headers* would occur *concurrently*, at

least because time is of the essence, and this processing would occur *in each of a*

plurality of protocol stack layers. *See* [13.3]; Ex.1003, ¶124.

12.   Claim 12

[12.0] *The method according to claim 1, wherein said concurrently writing step is performed within a physical layer utilizing a direct memory access (DMA) device.*

Cornett's NIC includes "a physical layer device (PHY)," the claimed

decoder. Ex.1005, [0079]; *see* [13.1]. Cornett's NIC *utiliz[es]* the DMA engine,

the claimed *direct memory access (DMA) device*, to write packet headers into

memory. Ex.1005, [0074]; *see* [13.2]. It would have been obvious to a POSITA

that Cornett's PHY uses Cornett's DMA engine to perform the *concurrently*

writing step. *See* [13.3]; Ex.1003, ¶127.

Cornett's PHY receives packets from the communications medium, i.e., the

physical layer, translates them, and causes storage of the packet's headers into

memory. Ex.1005, [0079]. It would have been obvious to a POSITA that Cornett's

PHY interfaces with the physical layer and higher layers of a protocol stack at least

because it is "coupled to a media access controller (MAC)…that performs 'layer 2' operations (e.g., Ethernet frame handling.)" Ex.1005, [0079]. A POSITA would have therefore understood that Cornett's PHY is operating *within a physical layer*. Ex.1003, ¶128.

XIII. GROUND 2: CORNETT-PAATELA-NELSON-RUSSELL RENDER OBVIOUS CLAIM 8

If the Board finds that Cornett-Paatela-Nelson does not adequately render claim 8 obvious, claim 8 is alternatively rendered obvious by Cornett-Paatela-Nelson-Russell for the additional reasons articulated below. Ex.1003, ¶140.

A.   Summary of Russell

Russell notes that several different schemes have been proposed to reduce latency between the time a packet is received and the time the packet is processed. Ex.1025, 1:33-35. Russell addresses this recognized latency issue by proposing the use of a processing system to improve packet processing. Ex.1025, Abstract, 3:13-14. The disclosed processing system "logically and physically separate[s] packet header processing functions from packet data processing functions." Ex.1025, Abstract, 3:13-14. The advantage of performing these operations, according to Russell, is that "network handling operations and data processing operations" can be performed "substantially in parallel." Ex.1025, 1:59-61. The processing system

may be "a personal computer, a workstation, a server computer, a router, a peer

device or other network node." Ex.1025, 3:24-26; Ex.1003, ¶129.

In one of its embodiments, packet parser 30 "parses" received packets and

"directs a portion of the packet header to first header memory 58 and directs the

remaining portion of the packet header to second header memory 60." Ex.1025,

5:7-10. The header information stored in memories 58 and 60 is then processed by

first and second header processors 54 and 56 "in accordance with one or more

network protocol handling operations[.]" Ex.1025, 5:22-27; Ex.1003, ¶¶130-131.

This is shown in Fig. 5:

Ex.1025, Fig. 5 (annotated)

IPR2024-01281 Petition
Inter Partes Review of U.S. 7,738,471
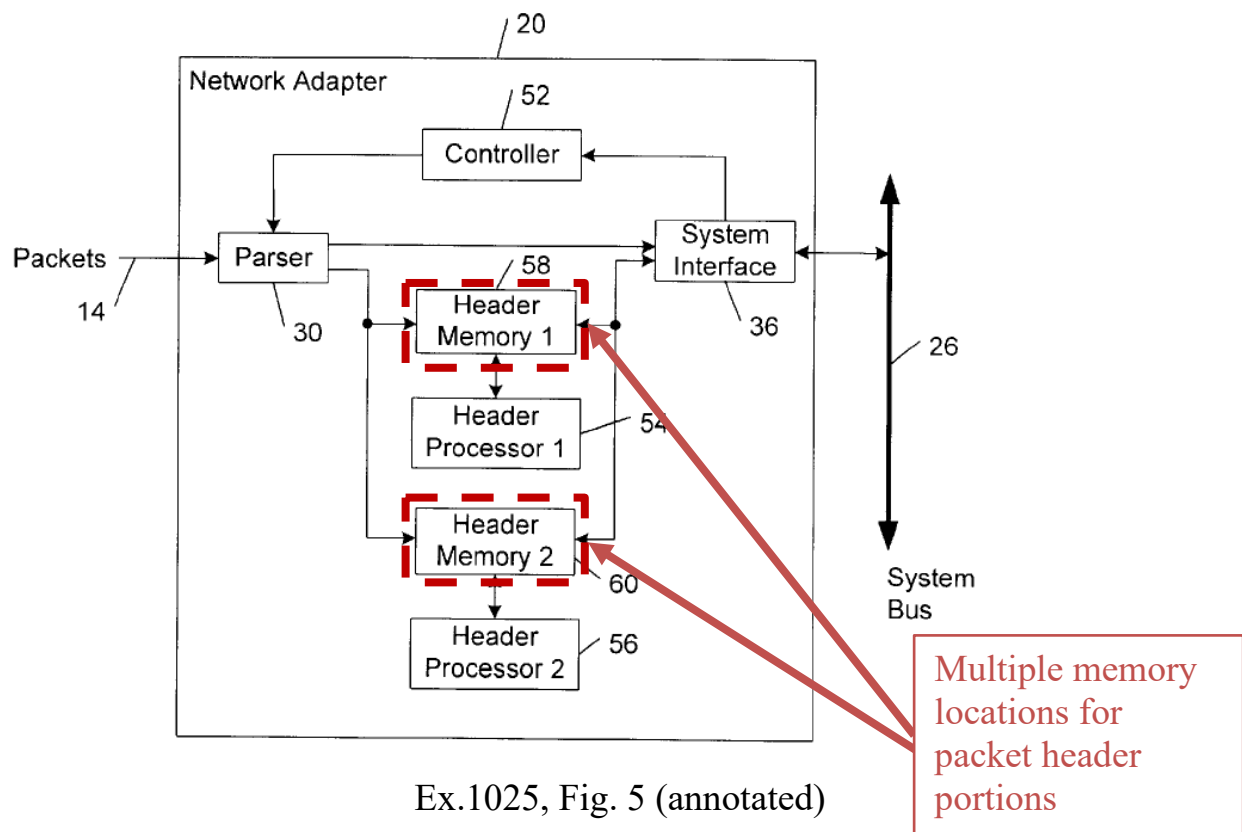
Russell and the '471 patent are both in the field of packet-based network communications and they both are specifically directed to the same topic: accelerated packet header processing. *See* Ex.1001, 1:8-11 ("invention relates to a method and apparatus for high speed protocol header processing[.]"); Ex.1025, Abstract, 1:33-35 (invention "provide[s] improved packet processing results" and "reduce[s] latency between the time a packet is received and the time the packet is processed."); Ex.1003, ¶132. Russell is also reasonably pertinent to the problem faced by the inventor of the '471 patent: how to improve packet processing results by processing packet headers individually. *See* Ex.1001, 3:32-34 ("concurrently processing the headers in each of a plurality of protocol stack layers."); Ex.1025, claim 8 at 8:7-9 ("two or more adapter processors each operable to process a respective one or more stored header components substantially in parallel."); Ex.1003, ¶132.

B.   Reasons to Combine Cornett, Paatela, Nelson, and Russell

A POSITA would have combined Cornett, Paatela, Nelson, and Russell because they all aim to accelerate packet processing. *See* Ex.1005, [0097] (enabling accelerated packet routing decisions by moving data from one processing buffer to another using a DMA engine); Ex.1007, [0076], Abstract, [0079] (accelerating packet processing by moving each one of a packet's encapsulated headers into individual memory segments and processing each header); Ex.1006,

1:13-20, 2:3-10 (accelerating packet routing decisions by eliminating the need to read the entire packet frame from memory, making the decision based on the frame header only); Ex.1025, Abstract, 1:33-35, 3:13-14 (improving packet processing by reducing latency); Ex.1003, ¶133. As discussed above, there is motivation to combine where a known technique "has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way." *Intel*, 21 F.4th at 800.

A POSITA would have been motivated to combine Cornett, Paatela, Nelson, and Russell to arrive at the claimed invention and the combination would have been within the POSITA's skillset. Ex.1003, ¶134. The combination represents the use of known techniques (writing packet headers into individual memory segments to accelerate processing, per Paatela, writing an entire packet frame into one buffer and the packet headers only into another buffer in parallel to accelerate routing decisions, per Nelson, and processing multiple packet headers in parallel, per Russell) to improve a similar process (accelerated packet processing technique that stores packet headers separately from an entire packet—both header and payload—using a DMA engine, per Cornett) in the same way to yield predictable results. Ex.1003, ¶134. As discussed above, such non-benefits-based combinations are permissible. *Intel*, 61 F.4th at 1381.

IPR2024-01281 Petition
Inter Partes Review of U.S. 7,738,471

As described above, Cornett-Paatela-Nelson processes packets, uses a DMA engine to save each individual packet header in a separate segment of memory, and uses the DMA engine to write to multiple memory segments in parallel, i.e., concurrently. *See* Sections XII.D. Once the headers are stored, however, the timing of when each individual packet header is processed is not explicit in the combination. Ex.1003, ¶135. Russell discloses this obvious yet missing detail when it teaches that packet headers are processed in parallel, thereby improving packet processing and reducing latency. Ex.1025, claim 8 at 8:8-9; 1:33-35, 3:13-14; Ex.1003, ¶135. A POSITA would have therefore been motivated to combine Cornett, Paatela, Nelson, and Russell for the efficient data processing reasons discussed below.

A POSITA would have appreciated that Cornett, Paatela, Nelson, and Russell seek to accelerate packet processing. Ex.1005, [0097]; Ex.1007, [0076], Abstract, [0079]; Ex.1006, 1:13-20, 2:3-10; Ex.1025, Abstract, 1:33-35, 3:13-14; Ex.1003, ¶136. And, as explained in [13.3], a POSITA would have understood that time is of the essence in Paatela. Ex.1007, [0076]. Similarly, Russell seeks to "reduce latency between the time a packet is received and the time the packet is processed." Ex.1025, 1:33-35. Russell achieves this by separating "packet header processing functions from packet data processing functions," Ex.1025, Abstract, 3:13-14, and then processing header portions "substantially in parallel." Ex.1025,

64

claim 8 at 8:8-9; Ex.1003, ¶136. A POSITA would have thus looked to Russell because a POSITA would have appreciated that processing header portions in parallel would further help achieve the purpose of each reference: accelerated packet processing. Ex.1005, [0097]; Ex.1007, [0076], Abstract, [0079]; Ex.1006, 1:13-20, 2:3-10; Ex.1025, Abstract, 1:33-35, 3:13-14; Ex.1003, ¶136.

A POSITA would have had a reasonable expectation of success in the proposed combination. Ex.1003, ¶137. A POSITA would have appreciated that Paatela stores individual packet headers in respective, separate memory segments to make each packet header available for processing as quickly as possible. Ex.1007, [0076]. Cornett seeks to further accelerate packet processing by using a DMA engine to free up the host processor and move data more efficiently. Ex.1005, [0097]. Nelson seeks to process packets faster by writing headers and frames into separate buffers—in parallel—so that header information is made available for routing decisions without the need (and associated delay) to read the entire frame from the buffer. Ex.1006, 2:3-10. And Russell processes headers "substantially in parallel" to further improve packet processing and reduce latency. Ex.1025, claim 8 at 8:8-9, 1:33-35, Abstract; Ex.1003, ¶137. The combination would achieve accelerated packet processing that the references seek to achieve.

Moreover, a POSITA would have been familiar with the concept of processing header portions in parallel. Ex.1003, ¶138. This is not a new concept.

Lawton, for example, discloses a Programmable Header Translator that performed look-ups on any combination of layer 2, 3, and 4 protocol fields. Ex.1026, 3. Similarly, Oskouy teaches a process for in-line packet processing that involves "pre-process[ing]" packet headers to "evaluate L2 header data for errors and locate the start of the next layer header (704)" while the "L3 header data is evaluated in parallel for errors (708)," as well. Ex.1024, [0118]. A POSITA would have understood Russell to disclose the obvious yet missing details for how to further the Cornett-Paatela-Nelson combination's accelerated packet processing technique. Ex.1003, ¶138. The combination would have been within the skillset of the POSITA based on the teachings of the references. Ex.1003, ¶¶137-138.

The Cornett-Paatela-Nelson-Russell combination is based on the teachings of the references themselves and does not require physical incorporation of elements from Paatela, Nelson, and Russell into Cornett. Ex.1003, ¶139. A physical incorporation of the teachings would nevertheless be possible for a POSITA and result in the claimed invention. Ex.1003, ¶139. The physical incorporation would use Paatela's input controller and packet classification engine to "determine[] where one networking layer header ends and the next networking layer header begins." Ex.1007, [0077]. It would then use Cornett's DMA engine to move each individual packet header into a respective memory segment, per Paatela. Ex.1005, [0074], [0097]. The move operation would be performed in

66

parallel, per Nelson, to further accelerate packet processing. Ex.1006, 7:1-4. Once

the individual packet headers are stored, each would be processed using Russell's

header processors "substantially in parallel." Ex.1025, claim 8 at 8:8-9; Ex.1003,

¶139. It would have been obvious to incorporate Paatela's, Nelson's, and Russell's

teachings into Cornett's solution. Ex.1003, ¶139.

C.   Detailed Analysis

1.   Claim 8

[8.0] *The method according to claim 1, further comprising concurrently processing said plurality of headers in each of a plurality of protocol stack layers.*

The Cornett-Paatela-Nelson combination does not explicitly teach the timing

of when each individual packet header stored in a respective, separate segment of

memory, per Paatela, is processed. Ex.1003, ¶141. This is disclosed by Russell. In

one of its embodiments, Russell discloses a network adapter 20 that is provided

within Russell's processing system 10. Ex.1025, Figs. 1, 5. The network adapter

includes a packet parser 30, first and second header processors 54 and 56, and first

and second header memories 58 and 60. Ex.1025, 5:3-7. In operation, "packet

parser 30 parses a received information packet 14, and directs a portion of the

packet header to first header memory 58 and directs the remaining portion of the

packet header to second header memory 60." Ex.1025, 5:7-10; *see also* Ex.1025,

claim 8 at 8:4-7 ("is configured to parse the packet header into two or more header

components and to direct each header component to a respective memory address."). Once the header portions have been stored, "[f]irst and second header processors 54, 56…process packet header information stored in header memories 58, 60 in accordance with one or more network protocol handling operations **(**e.g., firewall filtering, load balancing, and TCP/IP or RTP protocol handing operations)." Ex.1025, 5:22-27. These processors are "each operable to process a respective one or more stored header components substantially **in parallel**." Ex.1025, claim 8 at 8:8-9; Ex.1003, ¶¶142-144.

As an initial matter, it would have been obvious to a POSITA that Russell's reference to packet's header portions is a reference to a packet's one or more headers. *See* Section IV (packets have multiple headers); Ex.1003, ¶142. A POSITA would have been motivated to combine Cornett, Paatela, Nelson, and Russell. *See* Section XIII.B; Ex.1003, ¶145. Each of Cornett, Paatela, and Nelson accelerate packet processing. Ex.1005, [0097]; Ex.1007, [0076]; Ex.1006, 2:3-10. Russell also aims to reduce packet processing latency by "logically and physically separat[ing] packet header processing functions from packet data processing functions," Ex.1025, Abstract, 3:13-14, and writing packet headers to separate memory locations and processing them "substantially in parallel." Ex.1025, 5:7-10, claim 8 at 8:8-9. A POSITA would have thus been motivated to combine Cornett, Paatela, and Nelson with Russell because Russell's parallel processing of packet

68

headers would further accelerate packet processing and, as Russell states, reduce latency between the time a packet is received and the time the packet is processed, which would also help attain the goals of Cornett, Paatela, and Nelson. Ex.1025, 1:33-35; Ex.1003, ¶145.

In the combination, as explained in [13.3], Cornett's NIC would cause its DMA engine to store the packet (header and payload) in a first buffer, the claimed *packet buffer memory*, and each individual packet header in a separate segment of memory, the claimed *respective protocol stack layer memory*, for processing, per Paatela. Ex.1005, [0081]; Ex.1007, [0077]; Ex.1003, ¶146. Cornett's NIC would cause the DMA engine to perform both storage operations in parallel, i.e., concurrently, per Nelson. Ex.1006, 6:66-7:4; Ex.1003, ¶146.

Once the individual packet headers are stored, each would be processed "substantially in parallel," i.e., *concurrently*, per Russell. Ex.1025, claim 8 at 8:8-9; Ex.1003, ¶146. As Russell explains that each packet header portion is processed "in accordance with one or more network protocol handling operations," Ex.1025, 5:22-27, and Russell's reference to a header portion is a reference to a packet's header, a POSITA would have understood that each of the *plurality of headers* are *process[ed]…in each of a plurality of protocol stack layers.* Ex.1003, ¶147.

XIV. DISCRETIONARY DENIAL IS INAPPROPRIATE

A.   No Basis for 35 U.S.C. §325(d) Denial

Cornett, Paatela, Nelson, and Russell were not cited or considered during

prosecution of the '471 patent. Discretionary denial under §325(d) is inappropriate

at least because the first part of the *Advanced Bionics* framework is not met.

Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH, IPR2019-

01469, Paper 6 (Feb. 13, 2020) (precedential).

B.   No Basis for 35 U.S.C. §314/*Fintiv* Denial

The Board should not discretionarily deny institution based on the co-

pending district court litigation involving Patent Owner and Petitioner (*Lionra*

Technologies Ltd. v. Cisco Sys., Inc., 2:24-cv-00097 (E.D. Tex.)). To reduce

overlap between the two proceedings, if the Board institutes an IPR trial on this

Petition, Petitioner hereby stipulates that Petitioner will not pursue in the co-

pending litigation the specific grounds asserted in this petition, or any other ground

that could have been reasonably raised in an IPR (i.e., grounds that could have

been raised under §§102 or 103 on the basis of prior art patent or printed

publications). *See Sotera Wireless, Inc. v. Masimo Corp.*, IPR2020-00109, Paper

12, 18-19 (Dec. 1, 2020). Pursuant to the Director's "Interim Procedure for

Discretionary Denials," the Board should not discretionarily deny institution in

view of Petitioner's *Sotera* stipulation. EX1021, 3 ("Consistent with *Sotera*

70

IPR2024-01281 Petition
Inter Partes Review of U.S. 7,738,471

Wireless, Inc., the PTAB will not discretionarily deny institution in view of parallel district court litigation where a petitioner presents a stipulation not to pursue in a parallel proceeding the same grounds or any grounds that could have reasonably been raised before the PTAB."), 7-8 (similar).

C.   No Basis for *General Plastic* Denial

The '471 patent has not been challenged in any prior IPR petition, so none of the *General Plastic* discretionary institution factors apply to this Petition. *See General Plastic Indus. Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19, 16 (Sept. 6, 2016) (Section II.B.4.i. precedential).

XV.   CONCLUSION

Petitioner has established a reasonable likelihood that the Challenged Claims are unpatentable.

Respectfully submitted,

Dated: August 26, 2024	/Theodore M. Foster/
HAYNES AND BOONE, LLP	Theodore M. Foster
2323 Victory Avenue, Suite 700	Lead Counsel for Petitioner
Dallas, Texas 75219	Registration No. 57,456
Customer No. 27683	

XVI. MANDATORY NOTICES UNDER 37 C.F.R. §42.8

A.   Real Party-in-Interest

Cisco Systems, Inc. is the real party-in-interest.

B.   Related Matters

The '471 patent is or was involved in the following cases:

Case Heading	Number	Court	Filed
Lionra Technologies Ltd. v. Cisco Sys., Inc.	2-24-cv-00097	EDTX	Feb. 13, 2024

EP2201740, a patent related to the '471 patent, is or was involved in the

following cases:

Case Heading	Number	Court	Filed
Lionra Technologies Ltd. v. Cisco Sys., Inc. and *Cisco Systems GmbH*	UPC_CFI_58/2024	Hamburg, DE, 1st Inst., Local Div.	
	ACT_7940/2024 (Infringement Action)		Feb. 14, 2024
	CC_33752/2024 (Counterclaim for Revocation)		June 6, 2024

C.   Lead and Back-up Counsel and Service Information

<u>Lead Counsel</u>

Theodore M. Foster	Phone: (303) 382-6205
HAYNES AND BOONE, LLP	Fax: (214) 200-0853
2801 N. Harwood Street, Suite 2300	ipr.theo.foster@haynesboone.com
Dallas, TX 75201	USPTO Reg. No. 57,456

<u>Back-up Counsel</u>

David L. McCombs	Phone: (214) 651-5533
HAYNES AND BOONE, LLP	Fax: (214) 200-0853
2801 N. Harwood Street, Suite 2300	david.mccombs.ipr@haynesboone.com
Dallas, TX 75201	USPTO Reg. No. 32,271

Eugene Goryunov	Phone: (312) 216-1630
HAYNES AND BOONE, LLP	Fax: (214) 200-0853
2801 N. Harwood Street, Suite 2300	eugene.goryunov.ipr@haynesboone.com
Dallas, TX 75201	USPTO Reg. No. 61,579

Please address all correspondence to lead and back-up counsel. Petitioner

consents to email service as noted above and asks Patent Owner to do the same.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner hereby certifies, in accordance with and reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 13,004. Pursuant to 37 C.F.R. § 42.24(d), this word count excludes the table of contents, table of authorities, mandatory notices under § 42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing.

Dated: August 26, 2024   /Theodore M. Foster/
                                          Theodore M. Foster
                                          Lead Counsel for Petitioner
                                          Registration No. 57,456

CERTIFICATE OF SERVICE

The undersigned certifies that, in accordance with 37 C.F.R. § 42.6(e) and

37 C.F.R. § 42.105, service was made on Patent Owner as detailed below.

Date of service	August 26, 2024
Manner of service	Fed Ex
Documents served	Petition for *Inter Partes* Review Under 35 U.S.C. § 312 and 37 C.F.R. § 42.104 of U.S. 7,738,471; Petitioner's Exhibit List; Exhibits 1001-1026; and Petitioner's Power of Attorney.
Persons served	Atlantic IP c/o Lombard Geliebter LLP 1325 Avenue of the Americas, 28th Floor New York, NY 10019

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456